НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»
Науковий керівник кафедри
_____ Іван ДИЧКА
«___»_____2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення комп'ютерних та інформаційно-пошукових систем»

спеціальності 121 Інженерія програмного забезпечення

на тему: «Веб-додаток для просторового арбітражу криптовалют»

Виконав:
студент IV курсу, групи КП-61
Андрієвський Дмитрій Олександрович _____

Керівник:
Старший викладач кафедри ПЗКС, к.т.н.
Хіцко Яна Володимирівна _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н., доцент,
Онай Микола Володимирович _____

Рецензент:
Доцент кафедри СПіСКС, к.т.н.,
Бояринова Юлія Євгеніївна _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення комп’ютерних та інформаційно-пошукових систем»

«ЗАТВЕРДЖУЮ»

Науковий керівник кафедри

_____ Іван ДИЧКА

«___»_____2019 р.

ЗАВДАННЯ

на дипломний проєкт студенту

Андрієвському Дмитрію Олександровичу

1. Тема проєкту «Веб-додаток для просторового арбітражу криптовалют», керівник проекту Хіцко Яна Володимирівна, ст. викладач, к.т.н., затверджені наказом по університету від «25» травня 2020 р. №1181-с

2. Термін подання студентом проєкту «16» червня 2020 р.

3. Вихідні дані до проєкту: див. Технічне завдання.

4. Зміст пояснювальної записки:
   - аналіз існуючих рішень та актуальність теми;
   - обґрунтування вибору засобів реалізації;
   - розроблення веб-додатка;
   - аналіз розробленої системи.

5. Перелік обов’язкового графічного матеріалу:
   - алгоритм взаємодії графічних елементів (креслення);
   - схема бази даних (креслення);
   - діаграма варіантів використання (плакат);
   - діаграма послідовності проведення арбітражу (плакат).

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент		

7. Дата видачі завдання «31» жовтня 2019 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення літератури за тематикою проєкту	10.11.2019	
2.	Розроблення та узгодження технічного завдання	27.11.2019	
3.	Розроблення структури веб-додатку	14.12.2019	
4.	Підготовка матеріалів першого розділу дипломного проєкту	27.12.2019	
5.	Розроблення дизайну сторінок та графічних елементів	02.02.2020	
6.	Підготовка матеріалів другого розділу дипломного проєкту	19.02.2020	
7.	Програмна реалізація веб-додатку	12.03.2020	
8.	Тестування веб-додатку	20.03.2020	
9.	Підготовка матеріалів третього розділу дипломного проєкту	04.04.2020	
10.	Підготовка матеріалів четвертого розділу дипломного проєкту	24.04.2020	
11.	Підготовка графічної частини дипломного проєкту	12.05.2020	
12.	Оформлення документації дипломного проєкту	24.05.2020	

Студент Дмитрій АНДРІЄВСЬКИЙ

Керівник проєкту Яна ХІЦКО

АНОТАЦІЯ

Даний дипломний проєкт присвячений розробленню веб-додатку для просторового арбітражу на біржах криптовалют.

У роботі виконано порівняльний аналіз існуючих рішень для просторового та інших видів арбітражу, проаналізовано методи завдяки яким можна займатись просторовим арбітражем, обґрунтовано вибір технологій та допоміжних бібліотек серверної та клієнтської частин для реалізації даного веб-сервісу. Розроблений веб-сервіс надає користувачам інструмент для зручного просторового арбітражу на основі проаналізованих та зібраних даних з різних бірж криптовалют. Збір інформації проходить за рахунок автоматизованих інструментів. Оброблені дані подаються користувачеві у зручному вигляді. Також було розроблено зручний та зрозумілий інструмент для подальшого арбітражу на основі зібраних про біржі криптовалют даних.

В даному проєкті розроблено та досліджено: архітектуру серверної та клієнтської частини веб-сервісу, алгоритм автоматичного збору та обробки інформації, інструмент для зручного просторового арбітражу на основі зібраних даних, а також графічні елементи та дизайн веб-сторінок.

ABSTRACT

This diploma project is dedicated to the development of a web application for spatial arbitrage on cryptocurrency exchanges.

The project compares the existing solutions for spatial and other types of arbitration, analyzes the methods by which you can engage in spatial arbitration, substantiates the choice of technologies and auxiliary libraries of server and client parts for the implementation of this web service. The developed web service provides users with a tool for convenient spatial arbitrage based on analyzed and collected data from various cryptocurrency exchanges. Information is collected through automated tools. The processed data is provided to the user in a convenient form. A convenient and understandable tool for further arbitrage based on the data collected on cryptocurrency exchanges was also developed.

This project develops and researches: the architecture of the server and client part of the web service, the algorithm for automatic collection and processing of information, a tool for convenient spatial arbitration based on the collected data, as well as graphics and web page design.

ДП.045440-01-90 Веб-додаток для просторового арбітражу криптовалют.
Відомість проєкту

Позначення	Найменування	Кіл-ть	Примітка
	Документація проєкту		
ДП.045440-02-91	Веб-додаток для	5	
	просторового арбітражу		
	криптовалют. Технічне		
	завдання		
ДП.045440-03-81	Веб-додаток для	55	
	просторового арбітражу		
	криптовалют.		
	Пояснювальна записка		
ДП.045440-04-51	Веб-додаток для	4	
	просторового арбітражу		
	криптовалют. Програма		
	та методика тестування		
ДП.045440-05-34	Веб-додаток для	20	
	просторового арбітражу		
	криптовалют. Керівництво		
	користувача		

Позначення	Найменування	Кіл-ть	Примітка
ДП.045440-06-99	Веб-додаток для	1	
	просторового арбітражу		
	криптовалют. Алгоритм		
	взаємодії графічних		
	елементів. Схема		
	алгоритму		
ДП.045440-07-99	Веб-додаток для	1	
	просторового арбітражу		
	криптовалют. База		
	даних системи. Схема		
	даних		
ДП.045440-08-98	Веб-додаток для	1	
	просторового арбітражу		
	криптовалют.		
	Компакт-диск		

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«ЗАТВЕРДЖЕНО»

Науковий керівник кафедри

_____ Іван ДИЧКА

"___" _____ 2019 р.

ВЕБ ДОДАТОК ДЛЯ ПРОСТОРОВОГО АРБІТРАЖУ КРИПТОВАЛЮТАМИ

Технічне завдання

ДП.045440-02-91

«ПОГОДЖЕНО»

Керівник проєкту:

_____ Яна ХІЦКО

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Дмитрій АНДРІЄВСЬКИЙ

2019

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: веб-додаток для просторового арбітражу криптовалютами.

Галузь застосування: інформаційні технології.

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на дипломне проєктування, затверджене кафедрою програмного забезпечення комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім. Ігоря Сікорського).

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для використання сучасними людьми в якості інструменту, який полегшує процедуру просторового арбітражу та автоматизує прості процеси з метою надання користувачеві можливості зекономити свій час та уникнути помилок пов'язаних з людським фактором.

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Веб додаток для просторового арбітражу криптовалют повинен забезпечувати наступні основні функції:

- можливість тримати кошти в декількох доступних валютах;
- можливість при потребі вивести кошти з рахунку;
- можливість вибору по якому контракту проводити арбітражні операції;
- можливість самому обирати біржі за якими здійснювати просторовий арбітраж;
- можливість написати у службу підтримки та отримати відповідь.

Розробку виконати мовою програмування JavaScript з використанням шаблону проектування MVC.

Додаткові вимоги:

– наявність хедеру та футеру веб додатка;

– наявність анімованих кнопок;

– наявність системи нотифікацій;

– дизайн сторінок з використанням сучасних стандартів веб-дизайну.

5. ВИМОГИ ДО ПРОЄКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проекту повинна бути розроблена наступна документація:

1) пояснювальна записка;

2) програма та методика тестування;

3) керівництво користувача;

4) креслення:

– «Алгоритм взаємодії графічних елементів»;

– «Схема бази даних».

6. ЕТАПИ ПРОЄКТУВАННЯ

Вивчення літератури за тематикою роботи 07.11.2019

Розроблення та узгодження технічного завдання 20.11.2019

Розроблення структури системи..... 10.12.2019

Розроблення дизайну та графічних елементів 28.01.2020

Програмна реалізація системи..... 15.03.2020

Тестування системи..... 22.03.2020

Підготовка матеріалів текстової частини проєкту 15.04.2020

Підготовка матеріалів графічної частини проєкту..... 10.05.2020

Оформлення технічної документації проєкту 01.06.2020

7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«ЗАТВЕРДЖЕНО»
Науковий керівник кафедри
_____ Іван ДИЧКА
«___» _____ 2020 р.

ВЕБ-ДОДАТОК ДЛЯ ПРОСТОРОВОГО АРБІТРАЖУ КРИПТОВАЛЮТ

Пояснювальна записка

ДП.045440-03-81

«ПОГОДЖЕНО»

Керівник проєкту:

_____ Яна ХІЦКО

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Дмитрій АНДРІЄВСЬКИЙ

2020

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

GUI – Graphical user interface.

HTML – Hypertext Markup Language.

IDE – Integrated Development Environment.

iOS – iPhone operating system.

JS – Java Script.

JSON – JavaScript Object Notation.

MVC – Model-View-Controller – архітектурний шаблон «Модель-Представлення-Контролер».

PHP – Hypertext Preprocessor.

QA – Quality Assurance.

QUIK (Quickly Updatable Information Kit) - програмний комплекс для організації доступу до біржових торгів. Складається QUIK з серверної частини і робочих місць (терміналів) клієнта, взаємодіючих між собою через інтернет.

SQL – Structured Query Language.

Арбітраж (валютний) – валютні операції, які здійснюються банками (арбітражерами) з метою вилучення прибутку з різниці валютних курсів однієї і тієї самої грошової одиниці на різних валютних ринках (просторовий арбітраж) або різниці в динаміці курсу (часовий арбітраж).

БД – база даних.

Валютний контракт (валютний ф'ючерс) – договір на купівлю-продаж валюти в майбутньому, за яким продавець приймає зобов'язання продати, а покупець - купити певну кількість валюти в майбутньому за встановленим на момент угоди курсу.

Коротка позиція – можливість виставити акції на продаж (на пониження), не володіючи при цьому самим активом.

ПЗ – програмне забезпечення.

ВСТУП

Торгівля вручну на фондовій біржі поступово відходить у минуле. На зміну йде прогрес інформаційних технологій. Ще існують успішні трейдери, які досягли високих результатів при здійсненні спекулятивних операцій на ринку цінних паперів. Але на їх торгові рішення впливає безліч психологічних факторів. Альтернативою людських емоцій на біржі можуть бути тільки торгові автомати, або, торгові роботи. Торговий робот (автомат) – це програмний комплекс, в який закладено алгоритм здійснення операцій на ринку цінних паперів. Автомати виключають будь-який вид ризику. У комп'ютерної програми відсутні емоції, властиві людині, а значить, прийняте програмою рішення є вірним в рамках заданого алгоритму, створеного людиною.

Даний проєкт присвячений розробці та створенню зручного і корисного з точки зору користувача веб-додатка для полегшення процедури просторового арбітражу. Просторовий арбітраж – один з найлегших з точки зору розуміння технології, та один з найбільш затратних по часу методів арбітражу. Його суть полягає у моніторингу цін в один і той самий час на різних ринках (біржах). На відміну від часового арбітражу, де можна лише прогнозувати та сподіватись на потрібну нам ціну, у просторовому арбітражу ми бачимо усі дані відразу. Проблема полягає лише у одночасному спостеріганні за великою кількістю ресурсів, і це місце яке можна покращити за рахунок технологій.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОБГРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЄКТУ

1.1. Загальний опис проблеми торгівлі на біржах

Біржа – юридична особа, що забезпечує регулярне функціонування організації ринку товарів, валют, цінних паперів, похідних фінансових інструментів. Зараз торги здебільшого проходять в електронному вигляді. На біржі існує торговий сервер, до якого підключені акредитовані брокерські компанії через спеціальні глобальні лінії зв'язку. Брокери в своїх інтересах або інтересах клієнтів виставляють в торгові системи заявки на покупку або продаж цінних паперів, валюти, товару. Таким чином, біржа надає можливість зустрітися покупцям і продавцям, але вже не в своєму будинку, а на одному сервері, і укладати угоди купівлі-продажу.

Інша важлива функція біржі – організація та гарантування розрахунків за угодами, забезпечення механізму «поставка проти платежу» (Delivery against payment). Майже усі біржі отримують комісійний збір з кожної заключеної угоди, і це основне джерело їх доходів. Іншим джерелом можуть бути членські внески, плата за доступ до торгів, продаж біржової інформації.

Залежно від торгованих активів біржі поділяються на такі види:

1. Товарні.
2. Фондові.
3. Валютні/криптовалютні.
4. Ф'ючерсні.
5. Універсальні.

У даній роботі розглядається тільки криптовалютна біржа.

Брокер на ринку валют – продавці, юридична особа, професійний учасник ринку цінних паперів, що має право здійснювати операції з валютами за дорученням клієнта і за його рахунок.

Найбільш поширеним видом на фінансових ринках є просторовий арбітраж. Простими словами його схема виглядає так: ціна на один і той же актив на двох біржах істотно відрізняється. Трейдер купує цей актив там, де дешевше, і продає його там, де він коштує дорожче. Різниця в ціні активу йде в прибуток трейдеру.

Варто відзначити, що просторовий арбітраж можливий виключно через низьку ефективність котирування активів в рамках окремої торгової площадки зокрема і всієї системи в цілому. Також існує ключова залежність: чим менше сама система централізована, тим гірше налагоджена взаємодія між біржами. За рахунок того, що в основі всієї системи криптовалют лежить децентралізація, це дає відмінну можливість для криптовалютного арбітражу.

1.2. Аналіз існуючих рішень для даної задачі

На даний момент існують декілька програм для просторового арбітражу. Всі ці програми можна розділити на два типа: програми-порадники, які не здійснюють торговельних операцій, а тільки радять, і програми-роботи, які здійснюють торговельні операції за заданим алгоритмом.

1.2.1. «Межбиржевой арбитраж криптовалют» від AlgoTrading.center

Програма із заданою частотою сканує котирування між криптобіржами, і шукає можливості для арбітражу. У програмі можна вибрати біржі, які сканувати, валюти, які дивитися, валюти, які виключити, а також вести статистику по арбітражним можливостям, щоб можна було знаходити максимальні розбіжності між котируваннями (рис. 1).

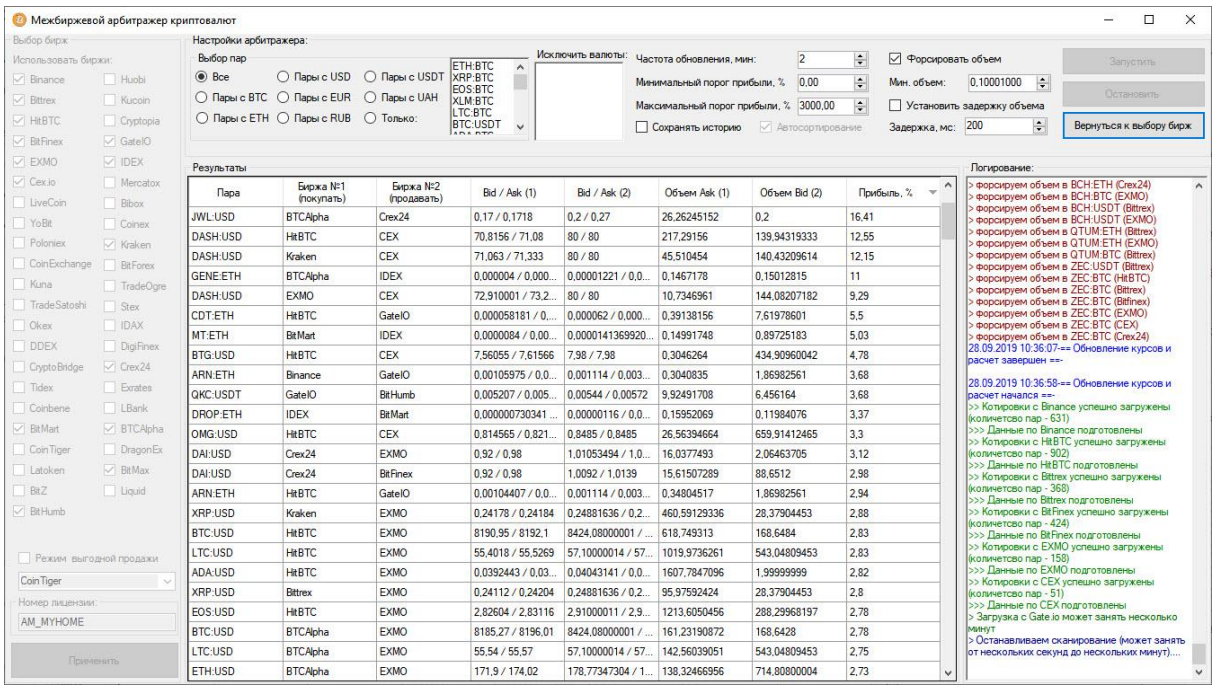


Рис. 1. «Межбиржевой арбитраж криптовалют»

Також програма дає змогу працювати з трьома видами арбітражу, серед яких:

1. Найпростіший вид – покупка + перепродаж криптовалют. Тобто, купуємо криптовалюту там, де вона дешевше, виводимо на біржу, де вона дорожче, і там її продаємо. Тим самим ми фіксуємо прибуток. В цій схемі є ризик (хоча, в загальному, в арбітражі вони відсутні) – це час. Тобто, поки гроші йдуть з однієї біржі на іншу, ціна може змінитися. Тому, якщо використовувати цей тип арбітражу, то потрібно обирати варіанти з великою розбіжністю.
2. Більш швидка схема – це маржинальний арбітраж криптовалют. У цій схемі, при розбіжності котирувань, ми купуємо на одній біржі, і продаємо на інший – маржинальній (тобто, де можна відкривати короткі позиції). В цьому випадку, криптовалюту не треба переводити між біржами, і таким чином, немає жодних ризиків.
3. І останній вид – це «одноногий» арбітраж. Суть його полягає в тому, щоб, при розбіжності котирувань, купувати криптопару,

яка дешевше, і продавати її, коли котирування зійшлися. Тим самим, ми відразу отримуємо математичну перевагу (у розмірі розбіжності котирувань), тобто, навіть якщо буде серія негативних і позитивних угод, то сумарно ми отримаємо прибуток за рахунок математичної переваги.

Отож дана програма завдяки обширній функціональності дозволяє займатись усіма типами просторового арбітражу.

1.2.2. WesternPips CryptoTrader 1.7

На вибір користувача в програмі реалізовані найпопулярніші на сьогодні криптовалютні біржі BITFINEX, BITSTAMP, BITTREX, BITMEX, GDAX, KRAKEN, POLONEX, LIQUI і інші (рис. 2). Нові біржі додаються регулярно, так само, як деякі біржі додають по запиту. Можна почати торгівлю як з однієї пари бірж на мінімальних обсягах лотів, так і використовувати всі наявні в програмі біржі.

Рис. 2. Westernpips Crypto Trader 1.7

В цілому, функціональні можливості такі самі, як і у попередній програмі, за винятком того, що було реалізовано технологію, яка дозволяє робити multi-leg арбітраж – тобто порівняння і арбітраж одночасно між усіма наявними біржами.

1.2.3. MetaStock

Найвідоміший програмний комплекс для технічного аналізу ринку. Являє собою новий стандарт для професійних трейдерів, що дозволяє проводити торговельні операції в реальному часі (рис. 3).

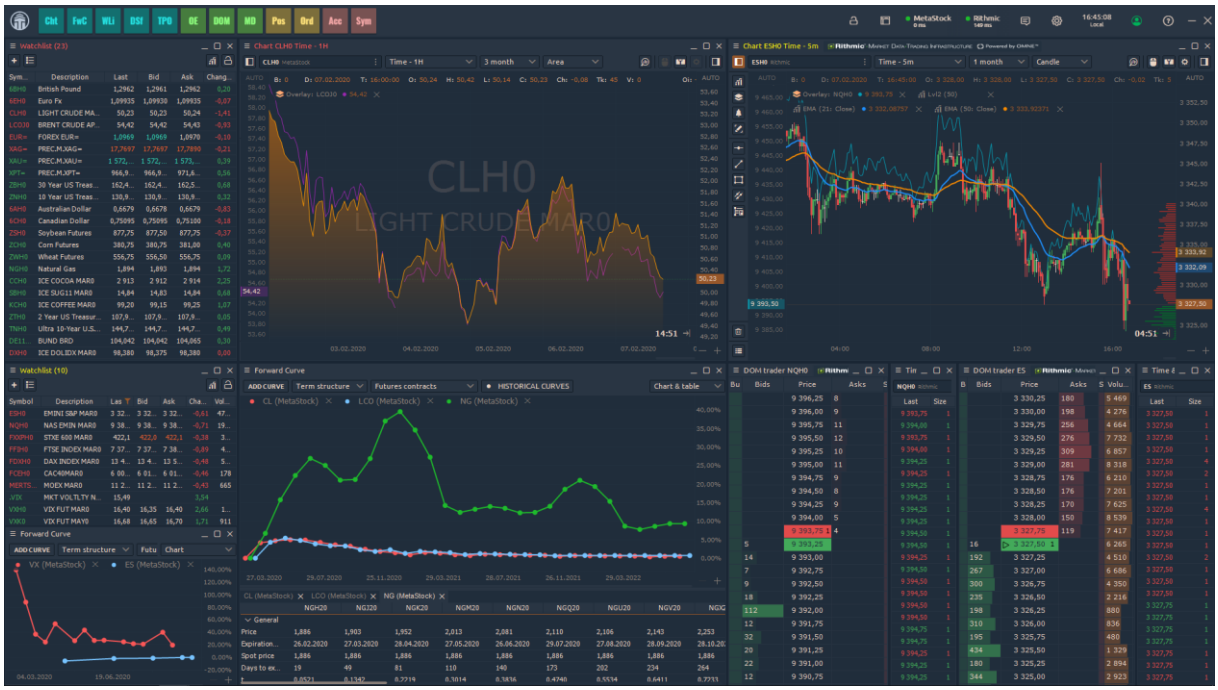


Рис. 3. MetaStock Xenith

За допомогою цього рішення трейдери можуть створювати, тестувати і повністю автоматизувати свої торгові системи, досягаючи тим самим максимальної прибутковості при проведенні операцій. На додаток вони зможуть користуватися всіма популярними можливостями MetaStock, такими як повна колекція торгових індикаторів, потужна пошукова система можливих персональних консультацій з експертами торгівлі та багато іншого.

Найзначніше нововведення в порівнянні з попередньою версією – кардинально перепрацьовано тестер систем, який став зручніше і функціонально насиченим. Це повнофункціональна професійна версія з підтримкою внутрішньоденних інтервалів і оновленням графіків у реальному часі.

1.3. Постановка задачі

«Межбиржевой арбитраж криптовалют» і WesternPips CryptoTrader 1.7 є програмами для професіоналів та їх інтерфейс може заплутати навіть досвідченого трейдера. Вартість використання програми «Межбиржевой арбитраж криптовалют» $375. Ціна на WesternPips CryptoTrader 1.7 близько $3000 (купити окрему програму неможливо та доводиться платити за весь пакет).

MetaStock безпосередньо пов'язаний з QUIK [1], ця програма обробляє експортовані дані та генерує сигнали. Програма створена, насамперед, для глибокого аналізу ринку. Звісно її можна налаштувати під просторовий арбітраж, проте для цього також потрібно бути досвідченим користувачем та трейдером.

Таким чином, актуальним є створенням зручного та простого веб-додатку для просторового арбітражу в реальному часі на біржах. При цьому до нього ставляться такі вимоги:

- ***Вимоги до інтерфейсу:***
  - адаптовність – простота переходу між різними девайсами, корректне відображення даних на екрані;
  - доступність – зрозуміла реакція та відповідь системи на різні запити;
  - зручність користування – можливість задовольнити потреби користувача за мінімальну кількість дій та простий інтерфейс;
  - гнучкість – адаптування інтерфейсу для того щоб вирішити

певну задачу.

- ***Апаратні вимоги:***
  - будь-який сучасний браузер(Google Chrome, Mozilla Firefox та інші);
  - доступ в мережу Інтернет (через Wi-Fi модуль або дротове з'єднання) для доступу до серверу веб-додатка.

- ***Вимоги до технологій:***
  - використання сучасних баз даних для швидкої обробки інформації щодо актуальних цін на біржах;
  - застосування «мікросервісної» розробки заради забезпечення стабільного функціонування системи та її компонентів, та у разі збою – швидкого виявлення проблеми та її усунення.

- ***Операційні вимоги***
  - надійність – шифрування паролів;
  - відновлюваність – використання кешування даних;
  - продуктивність – мінімальний час відклику при натисканні на екран у веб-додатку (2 секунди);
  - збереження даних – використання бази даних для збереження даних користувача;
  - керування помилками – коректне відображення даних при недопустимих запитах користувача.

1.4. Висновки до розділу

В результаті аналізу цієї предметної області можна зробити висновок, що існує необхідність створення веб-додатка для простого та швидкого початку опанування сфери трейдингу.

Можна сформулювати задачі, які повинен вирішувати веб-додаток моніторингу бірж криптовалют для подальшого просторового арбітражу:

- моніторинг значень покупки/продажу валют на багатьох біржах одночасно;
- надання інструменту для якомога швидкого виконання операцій купівлі/продажу на різних біржах.

Були проаналізовані наявні рішення та технології, які використовуються для просторового арбітражу. Необхідно використання «мікросервісної» розробки для зручного планування та розробки моніторингових інструментів, та застосування крос-браузерного способу розробки для забезпечення функціонування додатку на усіх сучасних веб-браузерах.

2. ОБГРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1. Обґрунтування вибору мови програмування

2.1.1. Мова програмування JavaScript

JavaScript, що часто скорочується як JS – мова програмування, яка відповідає специфікації ECMAScript. JavaScript високорівневий, часто компільований «на льоту» та мультипарадигмовий. Він має динамічну типізацію, прототипно-об'єктну орієнтацію та функції першого класу.

Поряд із HTML та CSS, JavaScript є однією з основних технологій всесвітньої павутини. JavaScript включає інтерактивні веб-сторінки і є невід'ємною частиною веб-додатків. Переважна більшість веб-сайтів використовують його для поведінки на стороні клієнта, а всі основні веб-браузери мають спеціальний механізм JavaScript для його виконання [2].

Основні IDE: Webstorm, Visual Studio Code.

Насамперед JavaScript було створено, щоб «оживити» веб-сторінки та додати їм динаміку. Програми написані на JavaScript часто називають скриптами. Вони вбудовуються в HTML і зазвичай виконуються одразу при завантаженні сторінки прямо у браузері. Вони не потребують компіляції та спеціальної підготовки перед виконанням. Це відрізняє JavaScript від мови програмування Java.

Так як і усі інші високорівневі мови програмування, JavaScript не надає розробнику доступу до низькорівневого коду та управління пам’яттю, що зменшує простір для помилок при написанні коду.

Сьогодні JavaScript може виконуватися не тільки в браузері, а й на сервері або на будь-якому іншому пристрої, який має спеціальну програму, що називається «рушієм» JavaScript. Всі сучасні браузери зараз мають вбудований рушій для цього під назвою «віртуальна машина JavaScript». Для розробки серверної частини додатку на JavaScript розробники зазвичай використовують програмну платформу Node.js.

Node.js – це крос-платформне середовище виконання JavaScript з відкритим кодом, яке виконує JavaScript за межами веб-браузера. Node.js дозволяє розробникам використовувати JavaScript для написання інструментів командного рядка та для сценаріїв на стороні сервера – від запуску скриптів на стороні сервера для створення динамічного вмісту веб-сторінки до відправки сторінки у веб-браузер користувача. Отже, Node.js представляє парадигму "JavaScript скрізь", що об'єднує розробку веб-додатків навколо однієї мови програмування, а не різних мов для скриптів на сервері та на клієнті [3].

Переваги JavaScript:

- Швидкість – JavaScript дуже швидкий, оскільки він виконується відразу в браузері клієнта. Поки користувач не відправляє додаткових запитів на сервер, який веде запуск, JavaScript не буде його сповільнювати своєю роботою.
- Простота – синтаксис JavaScript був створений під впливом мови Java, і легко вивчається, на відміну від інших популярних мов, такими як C, C ++.
- Популярність – JavaScript використовується скрізь в Інтернеті, а з появою Node.js почав все частіше використовуватись при серверній розробці. Є дуже велика кількість ресурсів для вивчення JavaScript, та з часом їх стає все більше.
- Взаємодія – на відміну від PHP або інших скриптових мов, JavaScript можна довільно вбудувати та запустити на будь-якій веб-сторінці. JavaScript також може використовуватися в багатьох інших системах через підтримку іншими мовами, такими як Pearl та PHP.
- Завантаження сервера – JavaScript є клієнтською надбудовою, тому він зменшує наплив на сервер в цілому, а прості додатки взагалі можна реалізувати без сервера.
- Універсальність – існує багато способів використання JavaScript

з Node.js. За його допомогою можна створити сервер на Express, використовувати базу даних документів на зразок MongoDB і також писати інтерфейс – таким чином можна розробити цілу програму, використовуючи лише JavaScript.

Недоліки JavaScript:

- захист на стороні клієнта – оскільки код JavaScript виконується на стороні клієнта, помилки та уразливості можуть використовуватися зловмисниками для їх дій. Через це деякі люди відключають виконання JavaScript сценаріїв у своїх браузерах.
- Підтримка браузера – хоча скрипти при виконанні на сервері завжди мають один і той же результат, різні браузери іноді розуміють код написаний на JavaScript по-різному. У наші дні різниці мінімальні, і це не є проблемою якщо проводити тестування скриптів у всіх основних браузерах.

2.1.2. Мова програмування PHP

PHP – популярна скриптова мова загального призначення, яка особливо підходить для веб-розробки. Спочатку вона була створена Расмусом Лердорфом у 1994 році. Реалізацію посилань на PHP зараз виробляє група PHP. PHP спочатку розшифровувався як Персональна домашня сторінка (Personal home page), але тепер це означає рекурсивний ініціалізм PHP: Hypertext Preprocessor [4].

Основні IDE: NetBeans, PHPStorm, Eclipse PDT.

PHP – це скриптова мова, яка зазвичай використовується у веб-розробці для розроблення серверної частини. Скриптові мови (сімейство мов програмування, включаючи PHP, а також такі мови, як JavaScript і Ruby) – це підмножина мов програмування, що використовуються для автоматизації процесів, які в іншому випадку повинні виконуватися покроково в коді сайту кожен раз, коли вони виникають.

Мови сценаріїв, такі як PHP, відрізняються від мов розмітки, таких як HTML та CSS. В той час як HTML і CSS визначають макет і зовнішній вигляд сторінок, скриптові мови кажуть статичній веб-сторінці (побудованій за допомогою тих же HTML і CSS) "робити" конкретні дії та змінювати свій вигляд в залежності від отриманих даних.

Переваги мови PHP:

- Крос-платформність – PHP-додаток можна запускати на різних платформах. Перевага PHP у тому, що розробник не повинен турбуватися на якій операційній системі працює користувач. PHP-код працює надійно і швидко скрізь.
- Простота використання – будь-які люди, які не знайомі з програмуванням, можуть легко навчитися писати на PHP. Синтаксис цієї мови програмування досить схожий на мову програмування C, проте є легшим та доступнішим для розуміння.
- Швидкість – дуже важливим аспектом веб-розробки є швидкість виконання коду. Веб-сайт який швидко завантажується завжди цінується людьми. Дуже велика кількість людей використовує PHP через те що він швидший ніж інші аналоги.
- Відкритий код та підтримка бібліотек – PHP розробляється та підтримується великим кластером розробників PHP, та це вплинуло на створення спільноти підтримки та розширеної колекції бібліотек. Крім того, PHP має величезні функціональні модулі колекцій, та деякі з модулів, доступних у PHP, включають графіку та PDF.
- Стабільність – PHP існує вже близько двадцяти двох років. За цей час багато розробників працювали над архітектурою. Багато помилок можуть залишатись непоміченими роками, тому у цьому аспекті великий час існування дуже позитивно вплинув на стабільність системи. Наразі мова програмування вважається дуже стабільною.

Недоліки мови PHP:

- слабка типізація – її буде важко використовувати для програмування великих додатків. Оскільки мова програмування не підтримує модульну розробку, великі програми, написані на PHP, важко систематизувати та зрозуміти.

Враховуючи переваги та недоліки розглянутих мов програмування, в цій роботі доцільно буде використовувати мову програмування JavaScript.

2.2. Обґрунтування вибору веб-фреймворку

2.2.1. ReactJS

React (також відомий як React.js або ReactJS) – це бібліотека JavaScript для побудови користувацьких інтерфейсів. Він підтримується Facebook та спільнотою окремих розробників та компаній. React можна використовувати як базу при розробці односторінкових або мобільних додатків. Однак React відповідає лише за надання даних у DOM, тому створення додатків на React зазвичай вимагає використання додаткових бібліотек для управління даними та маршрутизації. Redux [5] та React Router [6] є відповідними прикладами таких бібліотек [7].

React дозволяє розробникам створювати великі веб-додатки, які можуть маніпулювати даними не перевантажуючи веб-сторінку. Основна мета React – швидкість, масштабованість та простота. Фреймворк працює лише на користувацьких інтерфейсах у додатку. Це відповідає рівню «Представлення» в шаблоні MVC. Для розширення функціональності він може використовуватися у комбінації з іншими бібліотеками або фреймворками JavaScript, такими як AngularJS або Vue.js.

React замість звичайного JavaScript використовує JSX для програмування шаблонів. JSX – це простий JavaScript, який пишеться прямо в HTML, і використовує синтаксис тегів для візуалізації підкомпонентів. Синтаксис HTML обробляється у викликах JavaScript React Framework.

У React набір аргументів передається «будівнику» компонентів як властивості в HTML-тегах. Компонент не може безпосередньо змінювати ці властивості, але може передавати функцію зворотного дзвінка (callback), за допомогою якої і робляться модифікації веб-сторінки.

Переваги використання ReactJS:

- Простота – ReactJS простий для розуміння. Підхід розробки на основі компонентів, чітко визначений життєвий цикл та використання простого JavaScript роблять React дуже простим у вивченні та його підтримці. React використовує спеціальний синтаксис під назвою JSX, який дозволяє змішувати HTML з JavaScript, що значно пришвидшує розроблення додатку.
- Легкий у навчанні – кожен, хто має базові знання з програмування, може легко та швидко вивчити React, в той час як Angular та Ember називають "Доменною мовою", маючи на увазі що їх важко опанувати. Щоб почати писати на React, розробнику для початку потрібні лише базові знання CSS та HTML.
- Нативний підхід – React можна використовувати для створення мобільних додатків за допомогою вбудованої бібліотеки React Native. Тож одночасно ми можемо робити додатки для iOS, Android та веб-додатки.
- Прив’язка даних – React використовує односторонню прив'язку даних та архітектуру програми під назвою Flux та контролює потік даних до компонентів через одну контрольну точку – диспетчер. Через це простіше налагоджувати автономні компоненти великих програм ReactJS.

Недоліки використання ReactJS:

- Орієнтованість на «Представлення» – для розробки на ReactJS треба розроблювати свої рішення для рівнів «Модель» та «Контролер».

– Багато розробників не люблять документацію JSX React, через те що посібники важкі для розуміння новачками.
– Бібліотека важить доволі багато як для своїх функціональних можливостей.

2.2.2. AngularJS

AngularJS – це веб-фреймворк з відкритим кодом, тісно пов'язаний з JavaScript, підтримується Google, а також спільнотою осіб та корпорацій для вирішення багатьох проблем, що виникають при розробці односторінкових додатків. Він спрямований на спрощення як розробки, так і тестування таких додатків, забезпечуючи структуру для архітектури моделі-представлення-контролера (MVC) та моделі-представлення-представлення моделі (MVVM) на стороні клієнта, а також компоненти, які зазвичай використовуються у багатих Інтернет-програмах [8].

Веб-фреймворк AngularJS працює, попередньо інтерпретуючи веб-сторінку написану на HTML, на якій прописані додаткові спеціальні атрибути. Angular обробляє ці атрибути як аргументи щоб прив'язати вхідні або вихідні частини сторінки до моделі, яка представлена стандартними змінними JavaScript. Значення цих змінних встановлюється вручну в коді або отримується зі статичних або динамічних ресурсів таких як JSON.

Angular адаптує та розширює звичайний HTML-код для представлення динамічного контенту за допомогою двосторонньої передачі даних, що допомагає здійснювати синхронізацію моделей та представлень.

Переваги AngularJS:

– Open-source – AngularJS це фреймворк для архітектури MVC з відкритим JavaScript кодом. Оскільки він відкритий, то це забезпечує перевагу у легкій зміні вихідного коду. Ми можемо внести будь-які зміни у вбудовані компоненти системи, щоб задовольнити вимоги замовника.

– Фреймворк для односторінкового додатоку (SPA) – односторінковий додаток означає, що завантажується лише одна сторінка HTML, і подальше оновлення даних відбувається тільки на цій веб-сторінці. Оскільки фреймворк використовується для створення додатків на одній сторінці, він працює швидко і вважається «user-friendly».
– Не потребує глибоких знань – HTML, CSS та JavaScript, це все що потрібно для програмування на AngularJS. Якщо потрібно написати на Angular, це є простою задачею, оскільки HTML, CSS та JavaScript також є простими в навчанні.
– Легко розширювати та налаштовувати – завдяки певним вбудованим атрибутам фреймворк легко розширюється. Ці атрибути дозволяють розширити функціональність статичного HTML-коду, приєднавши до нього конкретну поведінку. Налаштування означає додавання або видалення функцій або функціональних можливостей, що робиться для задоволення конкретних потреб.
– AngularJs підтримує велика спільнота Google. Перевагами веб-сайтів, які підтримуються Google, є: регулярні оновлення, для віддалених користувачів можливість доступу в будь-який час і в будь-який час до корпоративної інтрамережі веб-сайтів, що підтримуються Google.
– Велика підтримка MVC – у багатьох інших фреймворках програміст повинен розділяти код на кілька компонентів архітектурного шаблону MVC. Після цього програмісту доводиться додатково кодувати, щоб поєднати код цих трьох частин. AngularJS це все робить автоматично. Фреймворк сам визначає та створює компоненти, поєднує код разом, а отже, економить час програміста.

– Легкий для тестування – Angular це фреймворк JavaScript, а JavaScript – динамічна мова. Оскільки фреймворк пропагандує вбудовану залежність компонентів, це полегшує тестування окремих компонентів коду.

Недоліки AngularJS:

– Менш захищений – немає авторизації та аутентифікації на стороні сервера. Авторизація означає надання дозволу на доступ до даних та ідентифікацію користувача шляхом перевірки даних. AngularJs не може надати такі функції, тому вважається мало захищеним.
– Використаня лише JavaScript – AngularJs повністю залежить від JavaScript. Якщо його відключити в браузері, сторінки на виході будуть мати дуже примитивний вигляд.
– Витік пам'яті – в JavaScript часто виникає проблема витоку пам'яті. Витік пам’яті це пам'ять, яку вимагає додаток, і яка через деякі фактори не повертається до пулу вільної пам’яті. Витік пам'яті призводить до різних інших проблем, таких як уповільнення роботи додатку, збої, висока затримка.
– Немає конкретного способу розробки – у AngularJS існує багато способів вирішення однієї і тієї ж проблеми, тому доволі важко передбачити, який спосіб є найбільш оптимізованим для виконання поставленого завдання.
– Не підтримується скрізь – Internet Explorer 8.0 не підтримує AngularJS.

2.2.3. Vue.js

Vue.js – це відкрите програмне забезпечення для JavaScript з парадигмою модель-представлення-представлення-модель для створення інтерфейсів користувача та односторінкових додатків. Він був створений

Еваном Ю, і його підтримують він та інші активні члени основної команди, що надходять від різних компаній, таких як Netlify та Netguru [9].

Vue.js має поступово адаптивну архітектуру, яка зосереджена на декларативному рендерингу та поєднанні компонентів. Розширені функції, необхідні для складних додатків, таких як маршрутизація, управління станом та інструменти збірки додатку, надаються через офіційно підтримувані бібліотеки та пакети, а Nuxt.js – одне з найпопулярніших рішень.

Vue.js дозволяє розширювати HTML за допомогою атрибутів HTML, які називаються директивами. Директиви пропонують функціональні можливості для програм HTML і бувають як вбудовані, так і визначені користувачем.

Переваги Vue.js:

- Доступність – Vue цілком простий для ще недосвідченого розробника. Маючи базові знання HTML, CSS та JavaScript, можна запустити проєкт на платформі Vue. Крім того, ви можете створити проєкт за хвилину, а не витрачати дні на це.
- Універсальність – Vue.js це екосистема, що розвивається на постійній основі, допомагає розробникам універсального або адаптивного характеру. Універсальність дозволяє Vue балансувати між бібліотекою і повнофункціональним фреймворком, що полегшує розробникам можливість створити якісну програму.
- Розмір – однією з найважливіших переваг використання Vue.js є його розмір, оскільки ви можете отримати готовий до виготовлення проєкт вагою всього 20 КБ після збірки.
- Продуктивність – малий розмір готового додатку призводить до більш швидкого виконання, а також пришвидшує розробку та дозволяє розробникам відокремлювати віртуальний DOM від компілятора. Більше того, якщо у вас мінімальний розмір

проєкту, вам не потрібно докладати додаткових зусиль для оптимізації.

– Простота інтеграції – будучи універсальним, Vue допомагає розробникам створювати якісні програми, які віходять інтегровними та легкими для тестування. Він може бути використаний як для створення SPA (односторінкового додатку), так і дуже складних веб-додатків, оскільки розробники отримують свободу інтегрувати менші частини до існуючої інфраструктури, не зачіпаючи всю систему.
– Масштабовність – Vue.js можливо використовувати окремо як бібліотека і як фреймворк. З другого боку, якщо потрібно здійснити будь-яку інтеграцію або розробити веб-додаток, Vue.js може запросто працювати з API REST, або будь-яким іншим методом отримання даних.
– Реактивність – передача даних між JavaScript та HTML реалізована через реактивну функцію, вбудовану у Vue.js, що робить можливим двосторонній зв'язок даних. Отже, коли відбуваються будь-яка зміна в даних, DOM також змінюється відповідно.
– Компоненти – повторне використання коду є однією з найкращих властивостей Vue.js, оскільки розробник отримує безліч компонентів та представлень, які можна легко інтегрувати в існуючий сервіс або додаток. Ця властивість дозволяє додавати стільки необхідних компонентів до вже наявних шаблонів, скільки буде потрібно.

Недоліки Vue.js:

– Мала кількість плагінів – плагіни бувають дуже корисні, оскільки вони зазвичай працюють з багатьма іншими інструментами, які полегшують та прискорюють розробку. Vue.js, на жаль, не має більшості загальних плагінів.

– Vue.js дуже швидко еволюціонує – те, що працює у Vue.js сьогодні, вже завтра може застаріти. Це найважливіший недолік Vue.js, оскільки для розробників це може помітно заповільнити розробку, через те що їм доводиться повторно перечитувати документацію фреймворку.
– Проблеми у використанні з iOS та Safari – якщо використовувати додаток Vue.js у старішій версії iOS та Safari, то можуть виникнуть різні проблеми та помилки, хоч вони і поступово виправляються.
– Відносно мале комьюніті – Vue.js було випущено лише у 2014 році, тому він все ще вважається новим. У результаті, він не настільки популярний у порівнянні з іншими фреймворками, такими як ReactJS та AngularJS. Крім того, оскільки Vue.js є розробкою китайської компанії, більша частина первинного коду написана китайською мовою, що створює деякі проблеми для розробників, адже більшість володіє лише англійською.
– Більш гнучкий, ніж потрібно – гнучкість це плюс, але це не завжди добре, якщо ви працюєте над якимись більш значущими проєктами, в яких беруть участь інші розробники, то це створить деякі проблеми. Надмірна гнучкість викликає більше помилок і нерегулярності в коді. Більше помилок та нерегулярності означає, що проєкти будуть затримуватися.

Отож проаналізувавши переваги та недоліки кожного веб-фреймворку та переглянувши функціональні вимоги до додатку, було вирішено використовувати ReactJS.

2.3. Обгрунтування вибору системи керування базами даних

2.3.1. PostgreSQL

PostgreSQL, також відома як Postgres – це вільна та відкрита система управління реляційними базами даних (RDBMS), що підкреслює

розширюваність та відповідність SQL. Спочатку вона мала назву POSTGRES, посилаючись на своє походження як спадкоємця бази даних Інгрес, розробленої в Каліфорнійському університеті Берклі. У 1996 році проєкт було перейменовано на PostgreSQL, щоб показати його підтримку SQL. Після огляду в 2007 році команда розробників вирішила зберегти ім'я PostgreSQL [10].

PostgreSQL оперує транзакціями з властивостями Atomicity, Consistentcy, Isolation, Durability (ACID), автоматично оновлюється представлення даних, матеріалізовані подання, тригери, зовнішні ключі та збережені процедури. Він призначений для роботи з різними навантаженнями, від окремих серверів до сховищ даних або веб-сервісів з багатьма одночасними користувачами. Це база даних за замовчуванням для сервера macOS, а також доступна для Linux, FreeBSD, OpenBSD та Windows.

PostgreSQL управляє своєю внутрішньою безпекою на основі ролей. Роль, як правило, вважається користувачем (роль, яка може ввійти), або групою (роль, яку мають інші ролі). Дозволи можуть бути надані або відкликані на будь-який об’єкт до рівня стовпця, а також можуть дозволити/запобігти створенню нових об'єктів на рівні бази даних, схеми або таблиці.

Переваги використання PostgreSQL:

- Хороша підтримка – система маємо живу спільноту професіоналів та ентузіастів PostgreSQL, до яких може звертатись кожен розробник та сприяти покращенню системи.
- Надійність та стабільність – багато користувачів в інтернеті пишуть, що PostgreSQL ніколи не виходив з ладу за кілька років високої активності.
- Розширюваний – вихідний код доступний для всіх безкоштовно. Якщо у розробників є необхідність будь-яким чином налаштувати або розширити PostgreSQL, вони можуть це зробити з мінімальними зусиллями і без додаткових витрат. Це

доповнюється спільнотою професіоналів та ентузіастів PostgreSQL по всьому світу, які також активно поширюють PostgreSQL щодня.

- Поперечна платформа – PostgreSQL доступний майже для кожної марки Unix, а сумісність з Windows доступна через систему Cygwin. Нативна сумісність для Windows також доступна з версією 8.0 і вище.
- Розроблено для середовищ з високим об'ємом – система використовує стратегію зберігання даних декількох рядків під назвою MVCC, щоб зробити PostgreSQL надзвичайно чутливим у середовищах з великим обсягом.
- Інструменти розробки та адміністрування баз даних GUI – існує багато високоякісних інструментів GUI, доступних для PostgreSQL як від розробників з відкритим кодом, так і від комерційних постачальників.

Недоліки використання PostgreSQL:

- Доступність для вивчення – є дуже достатня кількість матеріалів по PostgreSQL, проте все ж набагато менше ніж про інші системи керування SQL-запитами.

2.3.2. MySQL

MySQL – це система управління реляційними базами даних з відкритим кодом. Його назва - комбінація «My», ім'я дочки співзасновника Майкла Віденуса та «SQL», абревіатура для Structured Query Language.

MySQL – це вільне програмне забезпечення з відкритим кодом за умовами Загальної публічної ліцензії GNU, а також доступне під різноманітними власними ліцензіями. MySQL належить та спонсорується шведською компанією MySQL AB, яку придбала компанія Sun Microsystems (тепер корпорація Oracle). У 2010 році, коли Oracle придбав

Sun, Widenius відправив проєкт MySQL з відкритим кодом для створення MariaDB [11].

Основні функції MySQL:

- крос-платформне використання та підтримка;
- відповідність стандартам SQL;
- використання курсорів для полегшення обробки запитів додавання, вилучення і видалення записів бази даних;
- наявність статистики про сервер та запити;
- кількість рядків у таблицях до 50 мільйонів записів;
- кешування запитів;
- вкладені запити;
- вбудована підтримка репліка-сетів;
- підтримка Unicode;
- використання протоколу SSL для забезпечення додаткового захисту СУБД;
- повнотекстовий пошук та індексація;
- рідні системи зберігання: InnoDB, MyISAM, Merge, Memory, Federated, Archive, CSV, Blackhole, NDB Cluster.

Перевагами MySQL є простота, здатність до масштабування, швидкість роботи, якісне наслідування функціоналу від SQL, безкоштовне користування для некомерційних проєктів та надійність.

Основним недоліком MySQL є те що вона не спроможна одночасно виконувати багато процесів, що трохи понижує її ефективність серед інших інструментів.

Таким чином, проаналізувавши інформацію про систему PostgreSQL та СУБД MySQL, та порівнявши їх можливості, було виділено наступні переваги PostgreSQL над MySQL:

- PostgreSQL має достатні переваги, коли мова йде про підтримку CSV. Він надає різні команди, такі як «Копіювати» та «Скопіювати з», які допомагають у швидкій обробці даних. Він

також надає корисні повідомлення про помилки. Якщо є незначна проблема з імпортом, вона видасть помилку і пояснить у чому саме була проблема;

- PostgreSQL підтримує дуже велику кількість регулярних виразів у допомогу для аналітичної роботи. Звісно MySQL має подобні функціональні можливості у вигляді підряків, але вони не так добре працюють порівняно з регулярними виразами.

Враховуючи переваги та недоліки кожного способу зберігання та керування даними, в цій роботі буде доцільно використовувати систему для зберігання даних PostgreSQL.

2.4. Шаблон проєктування MVC

Model-View-Controller, зазвичай відомий як MVC – це архітектурний шаблон проєктування програмного забезпечення, який зазвичай використовується для розробки інтерфейсів користувача, що розділяє відповідну логіку програми на три взаємопов'язані елементи. Це робиться для того, щоб відокремити внутрішнє представлення інформації від способів подання та прийняття інформації для користувача.

Шаблон проєктування MVC пропонує розділити додатки, користувацький інтерфейс і керування логікою розроблюваної системи на три окремі компоненти (рис. 4): Модель, Представлення та Контролер, таким чином, що модифікація кожного компонента може здійснюватися незалежно.

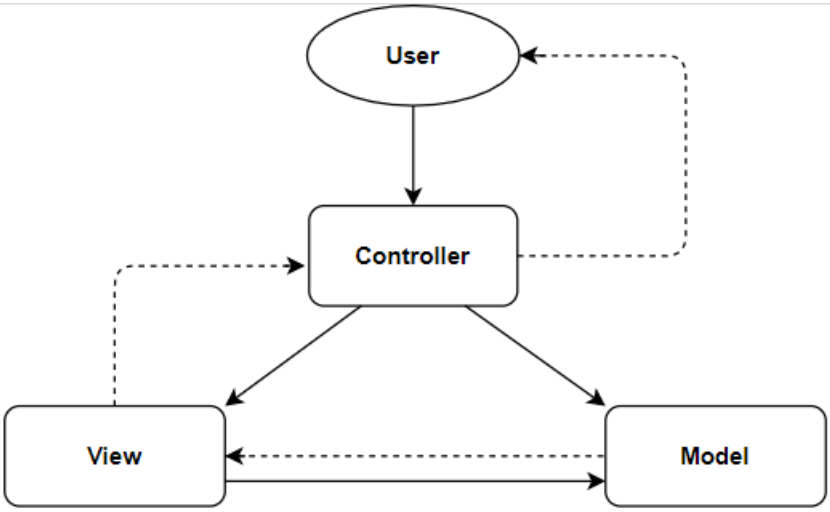


Рис. 4. Діаграма взаємодії між елементами шаблону MVC

Контролер – отримавши запит, контролер його аналізує і в залежності від результатів обробки може видавати наступні варіанти відповіді:

1. Надіслати відповідь про помилки (наприклад, при запиті неіснуючої сторінки віддає лише HTTP-заголовок «404 Не знайдено»).
2. Якщо отриманий запит визнано коректним, то, у залежності від того що він запитує (перегляд або на модифікацію даних), контролер викликає відповідний метод Моделі, такий як «Зберегти» або «Завантажити».

Модель – відповідає за представлення та доступ до даних додатка. В залежності від запиту, може віддати усі дані, частину даних, або ж видалити чи модифікувати дані. Дані всередині моделі можуть змінюватись, проте їх представлення повинно залишатись незмінним та незалежним. Як правило модель відповідає за роботу з БД.

Представлення – в залежності від отриманої від Моделі відповіді, Контролер вирішує, яке із доступних Представлень викликати для формування кінцевої відповіді на початковий запит: у випадку невдачі – Представлення для повідомлень про помилку, а у випадку успіху – Представлення для відображення запитуваних даних або будь-яких

повідомлень про успішність їх збереження(якщо початковий запит був на змінення даних). Також файли представлень часто називають шаблонами, а при використанні так званих шаблонізаторів роль Представлення грає сам шаблонізатор, а шаблони (тобто файли, що містять безпосередньо HTML-розмітку) у деяких фреймворках називають макетами.

У мові програмування JavaScript концепція шаблону проєктування MVC досягається за рахунок стандартних інструментів розробки. В результаті використання MVC програмний додаток стає надійним, зрозумілим та здатним до масштабування.

2.5. Висновки до розділу

В цьому розділі були розглянуті та проаналізовані основні інструменти для розробки веб-додатків. Отримана інформація була систематизована з метою використання в подальшій роботі для визначення конкретних засобів реалізації, що будуть використовуватись при розробці додатку для просторового арбітражу на біржах криптовалют.

Для розробки веб-додатку було вирішено обрати мову JavaScript, тому що:

- більш зрозумілий та швидкий для освоєння;
- більша сумісність з різними браузерами;
- велике комьюніті яке розвиває мову та розробляє різноманітні бібліотеки;
- дуже велика кількість бібліотек під різні задачі та потреби, та їх просте встановлення.

В якості системи керування базами даних вирішено обрати PostgreSQL через її швидкодію. Також серед усіх інших систем керування SQL-запитами, PostgreSQL зручніша для розуміння та має зрозумілішу структуру запитів.

Для побудови архітектури дипломного проєкту був обраний шаблон MVC, оскільки він забезпечує легкість підтримки програми, головні компоненти якої можливо змінювати незалежно один від одного.

3. РОЗРОБЛЕННЯ ВЕБ-ДОДАТКА

3.1. Структура програмного додатка

Веб-додаток було розроблено з виконанням усіх вимог описаних в підрозділі 1.3.

Користувач веб-додатка повинен зареєструватись, аби мати змогу щось робити – це необхідність так як основні дії на сайті можна робити лише зі своїми даними. Поки що при реєстрації користувачеві потрібно лише вказати свою пошту, пароль та підтвердження паролю. В наступних версіях продукту планується додати верифікацію даних по пошті. Для авторизації у системі потрібно ввести свій e-mail та пароль.

Авторизований користувач отримує доступ до основних сторінок додатка, таких як:

- «Баланси». На сторінці можна переглядати доступні кошти по кожній з доступних криптовалют, та у подальшому поповнювати або знімати кошти з балансів.
- «Контракти». Можливість побачити доступні для обміну напрямки валют, а також інформацію про кількість наявних бірж та їх комісію.
- «Трейдинг». У кожного «Контракта» є своя сторінка трейдинг, на якій проходять основні дії. Користувач може бачити вартість купівлі/продажу обраного курсу на різних біржах, і у подальшому обирати підходящі біржи для просторового арбітражу.

На сторінці трейдингу у простому для користувача вигляді реалізовано інструмент для проведення просторового арбітражу, який показано на рис. 5.

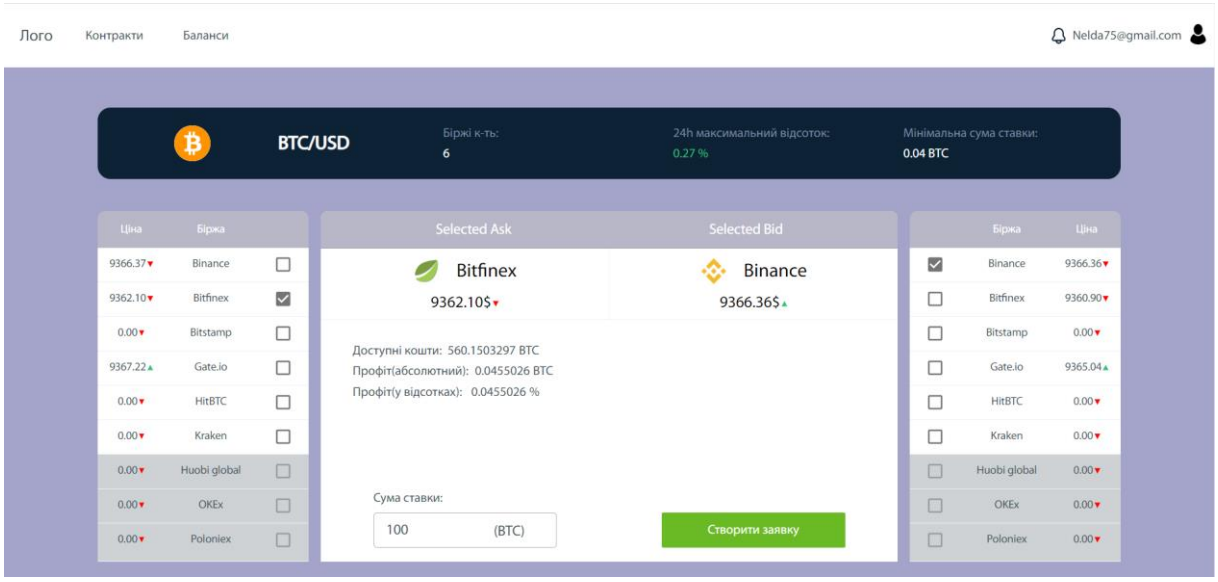


Рис. 5. Інструмент для проведення просторового арбітражу

У лівій частині користувач бачить за яку суму він може купити обрану криптовалюту на будь-якій із бірж, а у правій за яку суму продати. Обравши біржі з найбільшою різницею, користувач вводить суму та створює заявку на проведення арбітражу. Заявка через деякий час обробляється(здійснюються перекази між біржами), та користувач отримує свій дохід. На даному етапі проходить лише симуляція переказу коштів, адже для того аби проводити операції на біржах, треба оперувати реальними грошима.

Усі біржі обробляються запущеними на фоні невеликими модулями – так званими «воркерами». Це термінологія мікросервісної архітектури, логіка якої була запозичена для реалізації моніторингу бірж. Модулі, кожен з яких відповідає за окрему біржу, при запуску починають з певним інтервалом відсилати запити на API кожної з бірж та отримувати котирування по різним контрактам. Деякі біржі надають API для вільного використання, деякі можуть надати тестовий варіант по запиту(зазвичай якщо вказати використання для учбових цілей), а деякі взагалі його не мають, тому й у проєкті активовані не усі біржі, а лише частка.

Також у додатку реалізовано можливість звернутися у службу підтримки та отримати відповідь на своє запитання. Спілкування проходить

у реальному часі за допомогою додаткового модуля через який адміністратор, розробник, або у подальшому працівник сайту, може відповідати користувачу.

3.2. Аналіз функціональних вимог до програмного додатка

Аналіз функціональних вимог до веб-додатку наведено на таблиці 1.

Таблиця 1

Функціональні вимоги до веб-додатку для просторового арбітражу на біржах криптовалют

Код вимоги	Назва вимоги	Пріоритет вимоги	Коментар
01	Реєстрація в системі	1	З обов’язковим вказанням e-mail, паролю та підтвердження паролю
02	Авторизація в системі за допомогою e-mail та паролю	1	
03	Алгоритми моніторингу бірж криптовалют	1	
04	Створення заявки на купівлю-продаж по обраному курсу	1	
05	Можливість одночасного перегляду інформації по одному курсу на всіх біржах	1	
05	Перегляд доступних балансів користувача	2	

Продовження табл. 1

06	Зачислення коштів на баланс	3	Поки що усі «кошти» штучні
07	Виведення коштів з балансу	3	
08	Можливість написати у службу підтримки	3	
09	Можливість переглядати інформацію по всім заявкам на купівлю/продаж	2	
10	При проведенні арбітражної операції продажу/купівлі перевірка балансу користувача на достатність коштів	1	
11	Можливість розробнику/адміністратору ресурсу відключати певні контракти	2	Таким чином додається гнучкість контролю над додатком у разі якихось непередбачених випадків або збоїв у системі

3.3. Аналіз нефункціональних вимог до програмного додатка

Аналіз нефункціональних вимог до веб-додатка наведено на таблиці 2.

Таблиця 2

Нефункціональні вимоги до веб-додатку для просторового арбітражу на біржах криптовалют

Код вимоги	Назва вимоги	Пріоритет вимоги	Коментар
01	Валідація введених при авторизації даних	1	З обов’язковим вказанням e-mail та паролю

Продовження табл. 2

02	Необов'язкові сповіщення про успішність арбітражних операцій	3	У правому верхньому кутку екрану
03	Швидкість закінчення операцій перепродажу валюти не більше ніж 5 секунд	2	
04	Весь контент повинен бути у відведеній для нього зоні екрану, посередині. Тоді як header та footer(відповідно верхня та нижня зона додатку) повинні залишатись незмінні	1	Header та footer потрібні для зручної навігації по сайту
05	Токен авторизації користувача спадає через тиждень для більшої безпеки користування додатком	2	Користувачу доведеться кожного тижня наново вводити свої дані для входу

3.4. Архітектура програмного додатка

3.4.1. Опис кореневих файлів та бібліотек для БД

Теця «pm2» відповідає за запуск скриптів(мікросервісів), які займаються моніторингом обраних у роботі бірж криптовалют. За допомогою бібліотеки «pm2» [12] скрипти можна запускати як «демони» [13], де вони будуть працювати у фоновому режимі, а також слідкувати за їх роботою та відловлювати помилки шляхом їх логування. Наявні файли: pm2-binance.js, pm2-bitfinex.js, pm2-bitstamp.js і т.п.

Теця «public/static» використовується для збереження картинок, шрифтів, стилів, логотипів та дозволяє надавати доступ до цих файлів користувачу веб-додатку, без доступу до усіх інших файлів.

- docker-compose.yml – файл з налаштуваннями для докеру [14]. Docker дозволяє запускати усі потрібні для роботи додатку сервіси однією командою, замість того щоб робити це все окремо.

Для роботи з використаною у розробці БД PostgreSQL було використано наступні бібліотеки:

- Sequelize – використовується для підключення БД до проєкту та синхронізації моделей описаних у додатку.
- Sequelize-auto – бібліотека, яка автоматично генерує таблиці(об'єкти системи), згідно до синхронізованих схем. Використовується для швидкого тестування системи.
- Joi – бібліотека для валідації записуваних у БД даних. Оскільки бібліотека Sequelize є доволі низькорівневою та не оброблює помилки, для безперервної роботи додатку було вирішено використати зовнішню валідацію даних. Для кожного запиту до БД треба описати Joi-схему, яка буде перевіряти валідність та наявність потрібних даних, та у випадку якихось помилок видавати оброблювану помилку не звертаючись до самого сховища.
- bcrypt – бібліотека для створення хешованих-паролів. Для безпеки додатку у БД повинні зберігатися тільки захешовані паролі. Ця бібліотека дозволяє дуже просто захешувати пароль, та потім порівнювати введений користувачем пароль зі збереженим хешом. Наразі обирає популярність та є альтернативою алгоритмам SHA-2 які вважаються застарілими.
- pgAdmin4 – фреймворк який надає GUI до PostgreSQL.

3.5. Реалізація шаблону проєктування MVC

3.5.1. Models

Для реалізації вимог до веб-додатка та коректної роботи з даними у БД, було створено наступні моделі:

- User – схема, яка представляє користувача у системі. Вона складається з наступних полів: електронна пошта, хешований пароль, та дата реєстрації. Поля не можуть бути пустими, тому

при створенні нового користувача обов’язкоко проводиться перевірка даних, чи не є серед них нульові (null). Використовується для авторизації користувача у системі та подальшої авторизації користувачів на ресурсі. Більшість моделей у додатку прив’язані до користувача.

- Currency – схема, яка представляє валюту, та використовується як допоміжна для зручного відображення у додатку. Моделі Balance, Market, Order мають посилання на дану модель. Модель складається з наступних полів – логотип, назва, скорочена назва, схожий тип. Коли користувач викликає моделі де фігурують валюти, вони дістають дані через посилання на модель валюти. Це реалізовано аби уникнути дублювання однакових полів та запису повторюваних даних до БД.
- Balance – схема, яка представляє роботу з балансами користувача. У баланса може бути лише один власник (користувач). Модель складається з наступних полів – доступні кошти, утримані кошти, кошти у користуванні, посилання на валюту, посилання на користувача. Ці поля стосуються саме користувача. Схема тісно пов’язанна з заявками на купівлю/продаж, адже перед створенням заявки спочатку йде запит до моделі балансу, та якщо поле доступних коштів менше запитаної суми, система не дасть створити заявку.
- Market – схема, яка представляє собою курси купівлі/продажу по яким користувач може займатись просторовим арбітражем. Модель складається з наступних полів – назва, посилання на валюту купівлі, посилання на валюту продажу, значення мінімальної суми заявки, кількість доступних бірж для обміну, максимальну комісію. Використовується для безпосередньо арбітражу та пов’язана з моделлю заявок.
- Exchange – схема, яка представляє собою біржу криптовалют яка

може бути використана у просторовому арбітражі. Модель складається з наступних полів – назва. Використовується для відображення на сторінці створення заявок та як ідентифікатор для даних отриманих з роботи моніторингових скриптів.

- Order – схема, яка представляє собою заявку на купівлю/продаж по курсу криптовалют. Модель складається з наступних полів – посилання на біржу купівлі, посилання на біржу продажу, посилання на курс обміну, сума прибутку, посилання на валюту, час створення заявки, час завершення дій купівлі/продажу по заявці, вартість купівлі, вартість продажу, статус заявки, сума заявки. Являється основною функціональною моделлю додатка. Поля часу створення та часу завершення дій заявки повинні для відслідковування затримок перегону валют між біржами, адже вартість валют може змінюватись кожну секунду, що може приводити до відмінного від задуманого результату. Якщо користувачем буде виявлено затримку на декількох заявках то арбітраж по обраним біржам слід призупинити.
- Ticket – схема яка представляє собою звернення до служби підтримки. Модель складається з наступних полів – посилання на користувача, назву, статус. Являє собою місток між користувачем та працівником ресурсу у якому ведеться комунікація.
- TicketMessages – схема яка представляє собою повідомлення у зверненні користувача. Модель складається з наступних полів – посилання на звернення, текст, поле-перевірка ким відправлене повідомлення(користувач чи адміністратор). В залежності від того ким відправлене повідомлення відбувається позиціювання його щодо користувача у елементі чату.

На рис. 6 наведено схему представлення сутностей у базі даних.

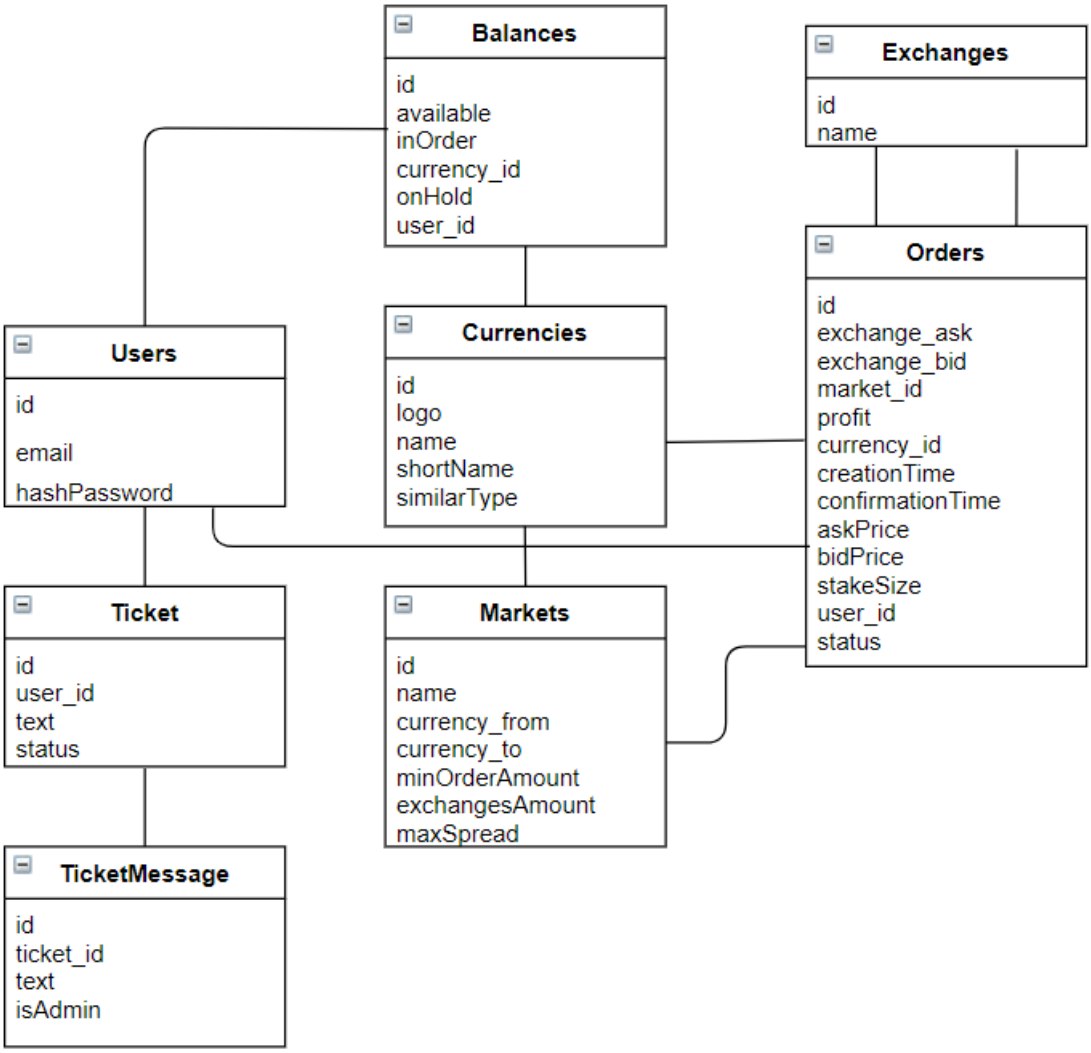


Рис. 6. Схема представлення БД

Основні сутності це «Користувачі» та «Валюти». На них ссилаються майже всі інші сутності у системі. Посилання на користувачів це обов’язкова міра, яка дозволяє системі розуміти якому саме користувачу належить сутність, та завдяки цьому далі оперувати цими сутностями у системі. Тоді як посилання на валюти зроблено спеціально, аби у подальшому, коли у системі з’явиться багато екземплярів сутностей, не потрібно було б змінювати кожен екземпляр окремо, а змінити лише сутність криптовалюти, а ці дані уже самі підтягуються до інших сутностей. Наприклад логотип криптовалюти зберігається лише у її сутності, у той час як ми можемо бачити його на різних сторінках котрі оперують іншими сутностями, але мають посилання на сутність криптовалюти.

Сутність «Заявка» має найбільшу кількість полів та посилань. Через посилання на «Контракти» система визначає по якому саме напряму здійснюється просторовий арбітраж; через «Валюти» та «Користувача» визначає який саме баланс треба поповнити по завершенню обробки.

3.5.2. Controllers

У веб-додатках роль контролерів відіграють так звані роутери – це модулі, які у залежності від запиту користувача вирішують яку саме модель та яку її функцію викликати, а також оперує отриманими даними.

Взаємодію у веб-додатку реалізовано через наступні контролери:

- Users_routes – роутер який відповідає за дії з користувачами. На створеній версії додатку це лише дії пов’язані з реєстрацією та авторизацією користувача. Роутер приймає дані введені у додатку, викликає валідацію даних, та якщо все добре, викликає функції створення користувача, або перевірку на наявність у сховищі згідно введених даних.
- Balances_routes – роутер, який відповідає за відображення балансів. Серед сценаріїв роботи даного роутера – повернення усіх балансів користувача та повернення балансу користувача по певній валюті. Якщо користувач відкриває сторінку балансів то йому повернуться усі його баланси. Контроллер повернення балансу по певній валюті створено для внутрішніх потреб, таких як перевірка чи вистачає у користувача коштів для створення певної заявки.
- Deposit_routes – роутер який відповідає за начислення коштів на баланс. Має усього один сценарій. Приймає суму поповнення, ідентифікатор користувача та ідентифікатор балансу який треба поповнити. Поки що це все робиться з «штучними» коштами які вручну створюються у базі.
- Withdraw_routes – роутер, який відповідає за виведення коштів з

балансу. Має усього один сценарій. Приймає суму виведення, ідентифікатор користувача та ідентифікатор балансу з якого будуть зняті кошти. Поки що все це робиться через «штучний» рахунок за допомогою сторонніх сервісів.

- Markets_routes – роутер, який відповідає за дії з напрямками обміну криптовалют. Має два сценарії. Повертає усі напрямки обміну криптовалют або ж дані по одному напрямку. Повернення усіх напрямків відбувається при відкритті користувачем сторінку з усіма курсами, де він може обрати потрібний йому курс для подальшого арбітражу. При переході на певний курс повертаються розширені дані для відображення сторінки обраного курсу, усіх доступних бірж та інструменту просторового арбітражу.
- Orders_routes – роутер, який відповідає за дії з заявками по продажу/купівлі криптовалют. Має два сценарії. Створення заявки по курсу та повернення уже створених заявок. Перед створенням заявки користувач повинен обрати біржі на яких він хоче продати та купити криптовалюту, а також суму яку він відводить на цю дію. Після цього заявка обробляється внутрішніми алгоритмами та підтверджується. Після підтвердження користувач може побачити заявку в історії заявок, за що відповідає другий сценарій який повертає усі підтверджені заявки та їх дані по ідентифікатору користувача.
- Tickets_routes – роутер, який відповідає за службу підтримки. Має декілька сценаріїв. Створення звернення, створення повідомлення та повернення даних звернення. Спочатку користувач повинен створити звернення та вказати його назву, після чого звернення буде створене у системі та адміністратор зможе відповісти на нього. Коли користувач або адміністратор створюють повідомлення, викликається відповідний сценарій,

який приймає текст повідомлення та ідентифікатор звернення до якого треба прив’язати повідомлення. При відкритті звернення викликається сценарій повернення даних звернення, таких як назва звернення, повідомлення прив’язані до звернення, та належність адміністратору.

3.5.3. Views

Графічний інтерфейс користувача реалізовано через наступні представлення:

- Header.js – клас у якому реалізовано хедер веб-додатку. Це невеличка зона наверху екрану, яка показується на кожній сторінці та є незмінною. Завдяки хедеру реалізовану зручну навігацію по сайту, завдяки чому користувач має змогу перейти на сторінки балансів, курсів обміну, служби підтримки, авторизації або реєстрації з будь-якого місця додатку.
- login.js – клас, який відповідає за авторизацію користувача у системі. Має форму у якій користувач може ввести свою пошту та пароль, і ці дані відправляться на сервер для подальшої обробки.
- signup.js – клас, який відповідає за реєстрацію користувача у системі. Складається з форми у якій користувач повинен ввести свою пошту, пароль та підтвердження паролю.
- balances.js – клас, в якому реалізовано показ усіх балансів користувача, повних даних по ним, а також доступ до функцій зачислення та виведення коштів. Відображається у вигляді таблиці.
- markets.js – клас у якому реалізовано показ усіх курсів обміну криптовалют та даних по ним, а також доступ до інструменту просторового арбітражу по певному курсу обміну. Відображається у вигляді таблиці.
- orders.js – клас, який відповідає за основні функції веб-додатку.

Відображає усі доступні біржі криптовалют, та вартість продажу/купівлі обраної криптовалюти у реальному часі. Дані постійно оновлюються. Також у даному класі реалізовано інструмент просторового арбітражу. Користувач повинен обрати біржу продажу криптовалюти із списку зліва, біржу купівлі криптовалюти із списку справа та ввести суму. Після цього заявка буде створена та оброблена системою. При завершенні заявки користувач зможе побачити свої заявки у списку який знаходиться нижче від вікна інструменту для просторового арбітражу.

- deposits.js – клас, який відповідає за начислення коштів на баланс. Реалізовано як модальне вікно у Представленні балансів. При натиску на кнопку відкривається модальне вікно де можна вказати на яку суму користувач хоче зробити поповнення, та адресу криптогаманця з якого зпишуться кошти.
- withdraws.js – клас який відповідає за виведення коштів з балансу. Реалізовано як модальне вікно у Представленні балансів. При натиску на кнопку відкривається модальне вікно де можна вказати яку суму користувач хоче вивести, та адресу свого криптогаманця на який відправити кошти.
- tickets.js – клас, який відповідає за спілкування зі службою підтримки. Реалізовано як список уже створених звернень та кнопкою при натисканні на яку відкривається модальне вікно створення нового звернення. При створенні нового звернення треба вказати його назву. При переході на сторінку звернення відбувається запит на всі повідомлення пов’язані з цим зверненням, та згідно до належності повідомлення адміністратору чи користувачу відбувається позиціонування повідомлень на екрані. Також внизу є поле для введення та відправки повідомлення.

3.6. Висновки до розділу

В цьому розділі було представлено загальний опис веб-додатку для просторового арбітражу криптовалютами.

Були представлені функціональні вимоги веб-додатку. Всі вимоги були реалізовані у системі.

Також були розглянуті використані технології, зокрема використання архітектурного шаблону Model-View-Controller (MVC), що є одним з найпопулярніших, найбільш використовуваних та ефективних шаблонів для розробки сучасних веб-додатків. У рамках архітектурного шаблону модель-представлення-контролер (MVC) програма поділяється на три окремі, але взаємопов’язані частини з розподілом функцій між компонентами, змінюючи які окремо один від одного робота системи залишається незмінною.

Для реалізації шаблону Model-View-Controller була використана IDE Microsoft Visual Code, яка є мультифункціональною середою розробки, та яку кожен розробник може налаштувати під себе для розроблення на більшості мов програмування, зокрема й на JavaScript.

4. АНАЛІЗ РОЗРОБЛЕНОЇ СИСТЕМИ

4.1. Тестування веб-додатка

Тестування програмного забезпечення – це один з етапів розробки ПЗ, який проводиться з метою отримання інформації про якість досліджуваного програмного продукту чи послуги [15].

Тестування програмного забезпечення також може забезпечити об'єктивний, незалежний погляд на програмне забезпечення, що дозволить бізнесу оцінити та зрозуміти ризики впровадження програмного забезпечення. Методи тестування включають процес виконання програми чи програми з метою пошуку програмних помилок (помилок чи інших дефектів) та перевірки відповідності програмного продукту для використання.

Для розробки даного додатка було використано Waterfall Model (каскадна модель), тому остаточне тестування проводилось лише 1 раз після закінчення етапу розробки. QA-спеціалістом виступив розробник даного додатку. Під час тестування було виявлено 25% помилок усього проєкту. Тестування проводилось за допомогою бібліотеки з відкритим сирцевим кодом mocha [16], яка дозволяє швидко та просто написати сценарії тестування та перейти безпосередньо до самого тестування додатку. Можливо якщо проводити тестування на етапах розробки можна зменшити цей процент, проте серед плюсів каскадної моделі є швидкість розробки, що й стало основним фактором при виборі цієї методології.

У даній роботі поетапно було проведено тестування основних складових веб-додатку:

4.1.1. GUI

GUI – у будь-якого тестованого додатка є зовнішній вигляд, тому тестування графічного інтерфейсу або просто зовнішнього вигляду – це

найперше, що ми можемо зробити. Порівняти його з вимогами і/або з макетом а також перевірити верстку.

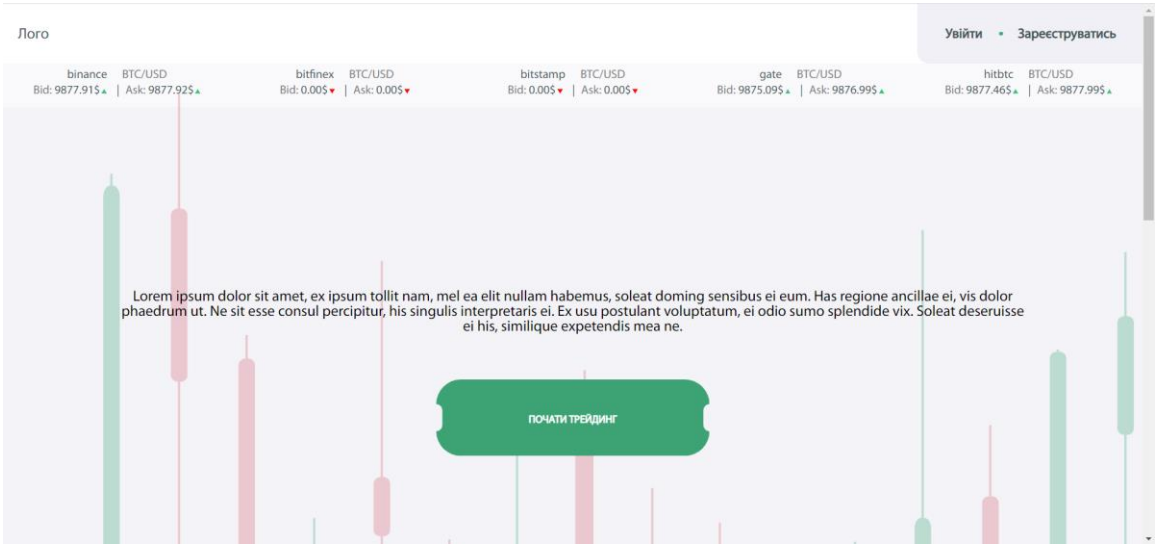


Рис. 7. Приклад графічного інтерфейсу веб-додатка

Верстка – розміщення елементів веб-додатку (зображення, текст, кнопки, відео) відповідно до макету або вимогам. Зазвичай перевіряються:

- наявність усіх елементів;
- їх розмір і колір;
- розташування відносно один-одного;

При тестуванні графічного інтерфейсу було виявлено та одразу виправлено декілька помилок пов’язаних з позиціями елементів, яких не було початково, та вони з’явились при додаванні нових елементів до інтерфейсу.

4.1.2. Функціональність

Від зовнішнього переходимо до внутрішнього – функціонального тестування. Якщо в тестуванні GUI ми перевіряли наявність і зовнішній вигляд елементів, то в функціональному тестуванні ми перевіряємо їх працездатність і взаємодію. Функціональне тестування було проведено методом «чорного ящика».

Тестування методом «чорного ящика», також відоме як тестування, засноване на специфікації або тестування поведінки – техніка тестування, заснована на роботі виключно з зовнішніми інтерфейсами тестованої системи [17].

Було знайдено 10 незацікавлених осіб які погодились протестувати даний додаток. Кожному з цих людей було надано лист з вимогами які вони мали протестувати. При користуванні сайтом «користувачі» заповнювали таблицю відповідно до функціональних вимог, відмічали їх реалізацію, а також описували знайдені помилки та свої побажання щодо можливого подальшого покращення додатку.

На рис. 7 наведено один із результатів тест-кейсу на перевірку доступних коштів при виведенні коштів з балансу.

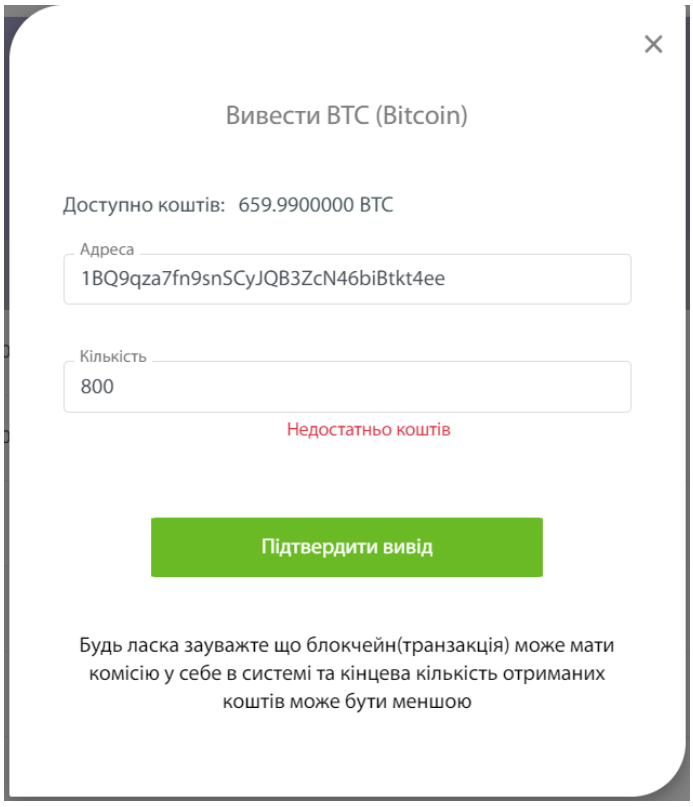


Рис. 7. Тест-кейс на виведення коштів

Загалом у процесі тестування було знайдено декілька як функціональних, так і нефункціональних помилок:

- при створенні заявки на продаж/купівлю валюти при недостатньому балансі – заявка все-одно створювалась та це приводило до краху системи;
- деякі орфографічні помилки у нотифікаціях та назвах елементів.

Перша помилка була допущена через те що не було реалізовано додаткову перевірку при створенні заявки – та після реалізації цієї перевірки проблему було усунено. Друга помилка сталася через людський фактор, адже розробкою займалась лише одна людина та таких помилок зазвичай можна уникнути коли є люди, які спеціально перевіряють даний вид помилок.

Після розробки додатку було виділено близько 20 годин на тестування веб-додатку, а також 30 годин на усунення виявлених помилок. Цього часу було достатньо аби отримати фінальну версію продукту.

4.2. Порівняння програмного додатка з аналогами

У першому розділі було дипломного проєкту було проаналізовано три аналоги розроблюваного веб-додатку та виявлено їх переваги та недоліки. Було виявлено основну проблему аналогів – складність у використанні. Згідно цього було розроблено вимоги до створюваного ПЗ які описані у кінці розділу. Розробка даного додатку велась згідно цих вимог та з урахуванням недоліків аналогів.

Отриманий кінцевий продукт відповідає наступним вимогам та перевагам перед існуючими аналогами:

- простота у використанні;
- швидкий перехід від реєстрації до арбітражу без необхідності поглибленого вивчення теми.

Також у кінцевому продукті було усунуто ще один великий недолік аналогів – складний та громіздкий інтерфейс. Його було покращено та

розроблено згідно до сучасних вимог до інтерфейсу. Також було досягнуто того що інтерфейс має бути інтуїтивно зрозумілим.

Ще одним нововведенням порівняно з конкурентами є наявність служби підтримки. Таким чином користувач має змогу звернутись до розробника/адміністратора ресурсу зі своїми проблемами та отримати швидку відповідь. Та ще це дає простір для подальшого розвинення додатку адже користувачі можуть вільно писати свої пропозиції які функції у перспективі можна додати до продукту.

Отож проаналізувавши отриманий додаток та порівнявши його з аналогами, можна стверджувати що він є цілком конкурентноспроможним. Він має менший функціонал, проте і створений для трохи іншої групи людей, та повністю задовольняє їх потреби.

4.3. Пропозиції для майбутнього поліпшення системи

Розроблений веб-додаток є мінімально-працюючим продуктом, що задовольняє критерії дипломного проєкту бакалавра, проте задля того аби стати комерційним та вийти на ринок існуючих функціональних можливостей недостатньо. Запити користувачів будуть зростати, а також із подальшим вивченням системи у користувачів будуть з’являтися потреби у нових функціях. Тому розроблений додаток можна вдосконалювати далі.

Після тестування додатку незацікавленими особами було отримано пропозиції для подальшого покращення системи у майбутньому:

- можливість реєстрації та авторизації у системі через Google, Facebook, та інші;
- додання можливості пошуку по біржах;
- додання можливості арбітражу по одній біржі але різними криптовалютами;
- додати можливість залишати відгуки про біржі криптовалют;
- додання системи нагадувань по пошті про завершені арбітражні заявки;

– реалізація інструмента для автоматичного створення заявок на продаж/купівлі при досягненні вартості певного курсу на біржі зазначеної заздалегідь суми;
– реалізація сторінки із статистикою по кожному курсу, де буде показано зміну його вартості продажу/купівлі у реальному часі, а також статистику по успішно обробленим заявкам інших користувачів;
– створення накопичувальної реферальної системи у веб-додатку та надання додаткових привілегій користувачам з певним стажем.

Також у розробленому програмному додатку є функція зворотнього зв'язку з розробником/адміністратором, де можна вказувати що завгодно від вказування на помилки сервісу, до пропозицій щодо покращення. Це є доказом того що продукт було створено з перспективою подальшого вдосконалення.

4.4. Висновки до розділу

У цьому розгляді було описано тестування розробленого веб-додатку для просторового арбітражу криптовалютами.

Було розглянуто методи тестування кінцевого програмного додатка. Тестування проводилось у декілька етапів. Спочатку було проведено ручне тестування основних функцій веб-додатка для перевірки коректності роботи розроблених модулів та алгоритмів системи. Після успішного проведення ручного тестування також було протестовано графічний інтерфейс додатку. Після цього етапу було проведено тестування функціональних вимог методом «чорного ящика». Було залучено 10 незацікавлених осіб які протягом певного часу мали протестувати основні та допоміжні функції веб-додатку. Після цього усі проблеми та помилки продукту було усунено.

Також було проведено порівняння отриманого веб-додатку з існуючими аналогами. Кінцевий продукт реалізовує виявлені переваги

перед існуючими рішеннями, а також покращує деякі недоліки які існують у конкурентів.

Було визначено пропозиції для подальшого покращення продукту та проаналізовані наявні функції, які забезпечують постійний зворотній зв’язок з користувачами та надання їм своєчасної допомоги.

ВИСНОВКИ

В результаті розробки було створено веб-додаток для просторового арбітражу криптовалют. Веб-додаток створено для починаючих трейдерів яким потрібен простий інструмент для того щоб почати займатись арбітражем, або для досвідчених користувачів котрим потрібно автоматизувати процеси для пришвидшення їх діяльності.

Було досліджено актуальність обраної тематики шляхом проведення опитування та аналізу використання аналогічних сервісів та їх популярності.

Був проведений аналіз існуючих програмних рішень шляхом порівняння переваг та недоліків аналогічних додатків для різних систем та цілей. Після цього було зроблено висновок щодо створення програмного рішення у вигляді веб-додатку та обраний спосіб розробки програмного рішення. Також, були визначені функціональні вимоги до розроблюваної системи, серед яких є апаратні вимоги, вимоги до інтерфейсу, вимоги до технологій.

Після аналізу можливих засобів реалізації було прийняте рішення розроблювати веб-додаток на основі NodeJs мовою програмування JavaScript, використовуючи СУБД PostgreSQL, і використовувати архітектурний шаблон MVC, який є найбільш популярним серед існуючих шаблонів проєктування веб-додатків.

Розроблений програмний додаток має наступні функції:

- реєстрація користувача за допомогою e-mail та паролю;
- авторизація користувача;
- роботу з балансами користувача – зачислення та вивід коштів;
- перегляд котирувань валют на доступних біржах;
- спрощене проведення арбітражних операцій за допомогою спроєктованого інструменту;

– можливість звернутись у службу підтримки та отримати відповідь від адміністратора ресурсу.

Веб-додаток розроблений в повному обсязі з відповідністю до розроблених вимог. Тестування веб-додатка виконано в повному обсязі згідно наявної методики тестування.

Розроблений веб-додаток допомагає користувачам економити свій час, спрощує процедуру проведення просторового арбітражу та дозволяє уникнути помилок пов’язаних з людським фактором.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Торговый терминал QUIK: [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.zerich.com/internet-trading/trade-systems/quik.html
2. JavaScript: [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/JavaScript
3. Node.js: [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://nodejs.org/en/
4. PHP: [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/PHP
5. Redux: [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://redux.js.org/faq/general
6. React Router: [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://reacttraining.com/react-router/web/guides/quick-start
7. ReactJS: [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://reactjs.org/
8. AngularJS: [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/AngularJS
9. VueJs: [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Vue.js
10. PostgreSQL: [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.postgresql.org/about/
11. MySQL: [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/MySQL
12. PM2 – Quick Start: [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://pm2.keymetrics.io/docs/usage/quick-start/
13. Daemon (computing): [Електронний ресурс]. – 2020 – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Daemon_(computing)
14. Use Cases | Docker: [Електронний ресурс]. – 2020 – Режим доступу до

ресурсу: https://www.docker.com/use-cases

15. Software testing: [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Software_testing
16. Mocha - the fun, simple, flexible JavaScript test framework: [Електронний ресурс]. – 2020 – Режим доступу до ресурсу: https://mochajs.org/
17. White/Black/Grey Box-тестирование: [Електронний ресурс]. – 2020 – Режим доступу до ресурсу: https://qalight.com.ua/baza-znaniy/white-black-grey-box-testirovanie/

ДОДАТКИ

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«ЗАТВЕРДЖЕНО»

Науковий керівник кафедри

_____ Іван ДИЧКА

«___» _____ 2019 р.

ВЕБ-ДОДАТОК ДЛЯ ПРОСТОРОВОГО АРБІТРАЖУ КРИПТОВАЛЮТ

Програма та методика тестування

ДП.045440-04-51

«ПОГОДЖЕНО»

Керівник проєкту:

_____ Яна ХІЦКО

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Дмитрій АНДРІЄВСЬКИЙ

2019

ЗМІСТ

1. ОБ’ЄКТ ВИПРОБУВАНЬ

Веб-додаток для просторового арбітражу криптовалют являє собою так званий сайт, створений на платформі NodeJS мовою програмування JavaScript з використанням шаблону проектування MVC.

2. МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

1) відповідність елементів сторінок вимогам до веб-додатку;
2) наявність доступу до процедури арбітражу;
3) зручність роботи з веб-додатком;
4) відповідність дизайну вимогам Технічного завдання.

3. МЕТОДИ ТЕСТУВАННЯ

Тестування виконується методом Black Box Testing. При цьому підході програмний продукт перевіряється на відповідність функціональним вимогам з точки зору зовнішнього світу, без урахування знань про те як насправді влаштована система. Тестування відбувається на рівні «системного тестування».

Використовуються наступні методи:

1) функціональне тестування;
2) тестування продуктивності програмного забезпечення, стабільності та Load testing (навантажувальне тестування);
3) тестування граничних значень;
4) тестування інтерфейсу.

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується за допомогою бібліотеки mocha.

Працездатність веб додатку для просторового арбітражу криптовалют перевіряється шляхом:

1) автоматичного динамічного тестування – прописані тест-кейси на введення граничних та недопустимих значень в поля де є певні обмеження;
2) динамічного ручного тестування на відповідність функціональним вимогам;
3) тестування додатка на різних браузерах та на різних комп’ютерах;
4) тестування при максимальному навантаженні на сервер;
5) тестування швидкодії при повільному з’єднанні з мережею інтернет;
6) тестування зручності використання;
7) тестування інтерфейсу.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«ЗАТВЕРДЖЕНО»
Науковий керівник кафедри
_____ Іван ДИЧКА
«___» _____ 2020 р.

ВЕБ-ДОДАТОК ДЛЯ ПРОСТОРОВОГО АРБІТРАЖУ КРИПТОВАЛЮТ

Керівництво користувача

ДП.045440-05-34

«ПОГОДЖЕНО»

Керівник проекту:

_____ Яна ХІЦКО

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Дмитрій АНДРІЄВСЬКИЙ

2020

ЗМІСТ

1. Опис структури мобільного додатка

Веб-додаток для просторового арбітражу криптовалют складається з мінімуму сторінок необхідних для того, щоб швидко розпочати займатись просторовим арбітражем. Додаток виконаний українською мовою.

Сторінки веб-додатка:

– початкова сторінка (при запуску додатка) має ціни контрактів по біржам, які оновлюються у реальному часі, а також список недавно завершених арбітражних заявок;
– сторінка реєстрації користувача;
– сторінка авторизації користувача;
– «шапка», яка має посилання на сторінки реєстрації та авторизації, а у зареєстрованого користувача ще на сторінки контрактів та балансів;
– «нижня шапка» з текстовим наповненням, створена для більшої презентабельності додатку;
– сторінка балансів користувача;
– сторінка доступних контрактів;
– сторінка створення арбітражних заявок по кожному контракту, містить також історію заявок користувача;
– сторінки для створення звернень у службу підтримки;
– сторінка окремого звернення (реалізована у вигляді чату).

Кожна сторінка містить верхню і нижню шапки, за допомогою яких користувач може переходити до усіх наявних сторінок додатка з будь-якої сторінки.

2. Опис вмісту головної сторінки мобільного додатка

Початкова сторінка додатка.

Початкова сторінка містить у собі додаткову інформацію для незареєстрованого користувача: ціни на контракти на біржах у реальному часі, та історію по останнім арбітражним заявкам. Також з цієї сторінки користувач може перейти до сторінки реєстрації або авторизації.

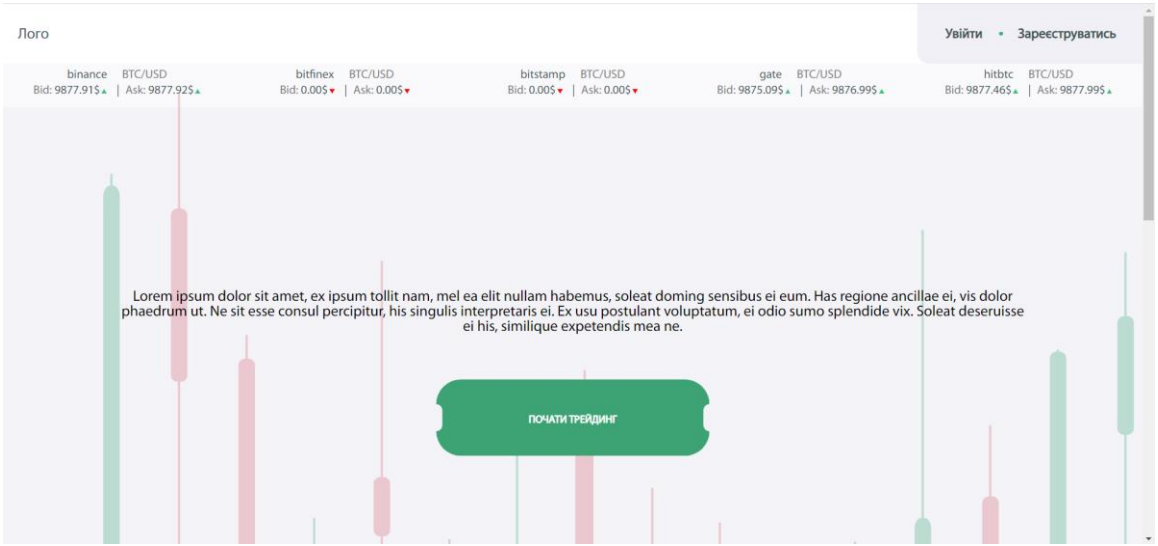


Рис. 1. Верхня частина головної сторінки

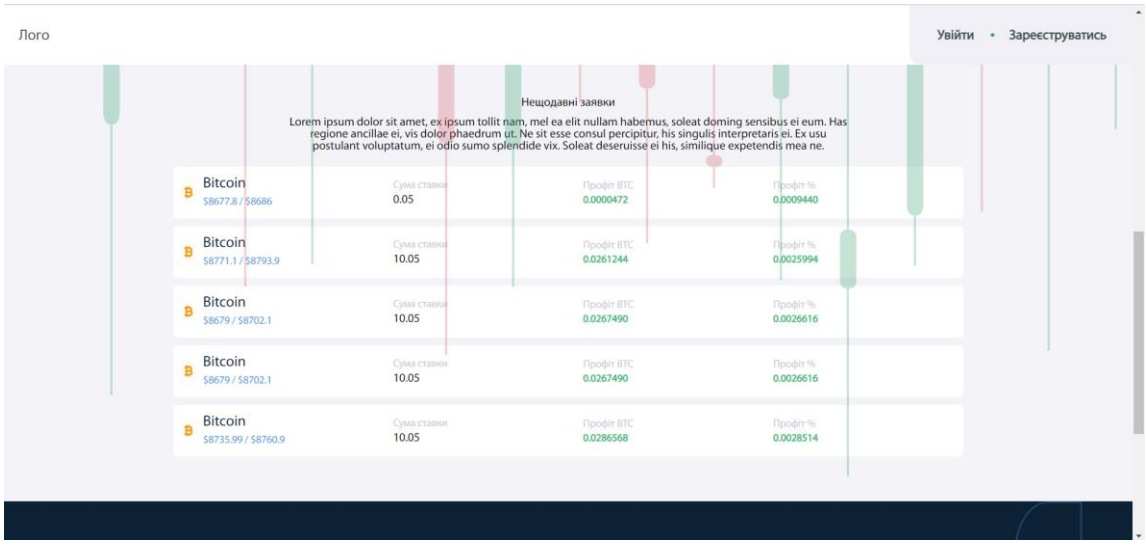


Рис. 2. Історія останніх арбітражних заявок

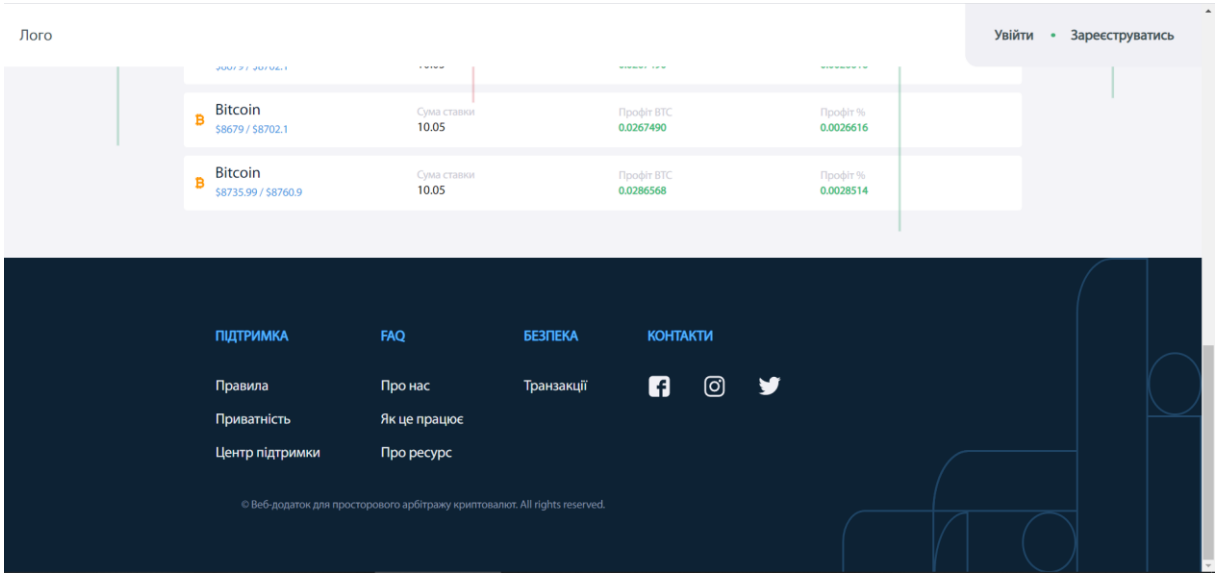


Рис. 3. «Нижня шапка» додатка

3. Процедура реєстрації та авторизації користувача

Процедура реєстрації.

При натисканні кнопки «Зареєструватись» (рис. 1), користувач переходить до сторінки реєстрації (рис. 4). Для того щоб зареєструватись користувач має ввести наступні дані – e-mail, пароль та повторити пароль. Введення даних до всіх полів оброблено згідно необхідної валідації. Наприклад, якщо користувач з введеним e-mail вже існує, то виникає помилка (рис. 5). Також створено валідацію пароля згідно до сучасних вимог (рис. 6, 7), та перевірка щодо коректного повторного введення паролю (рис. 8).

Після реєстрації користувач автоматично авторизується в системі.

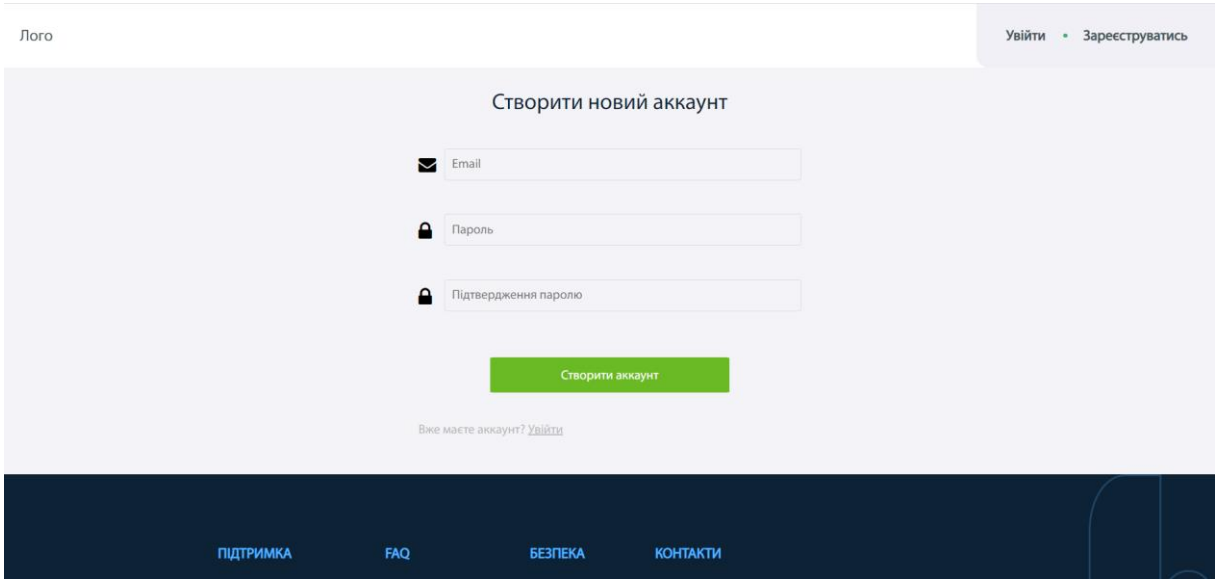


Рис. 4. Сторінка реєстрації користувача

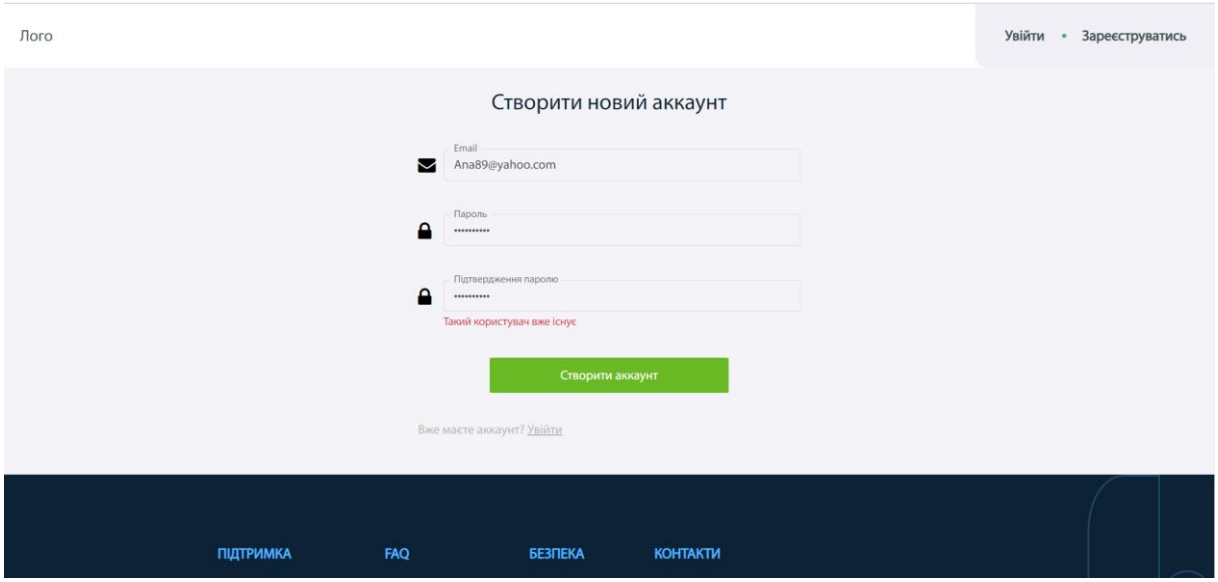


Рис. 5. Повідомлення про помилку на сторінці реєстрації

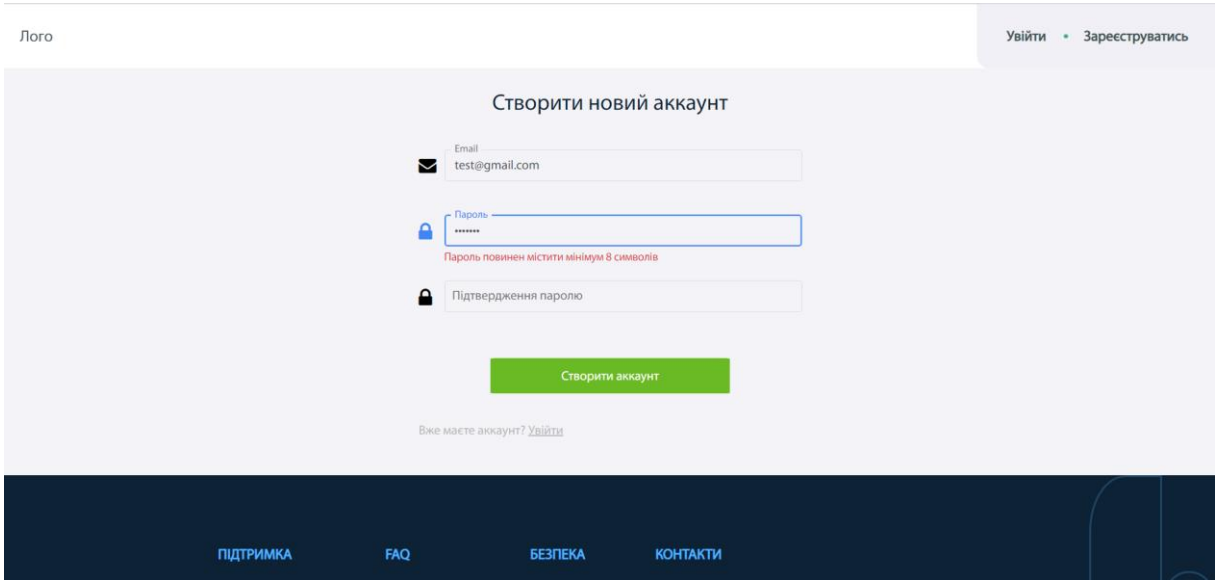


Рис. 6. Повідомлення про недостатню довжину паролю

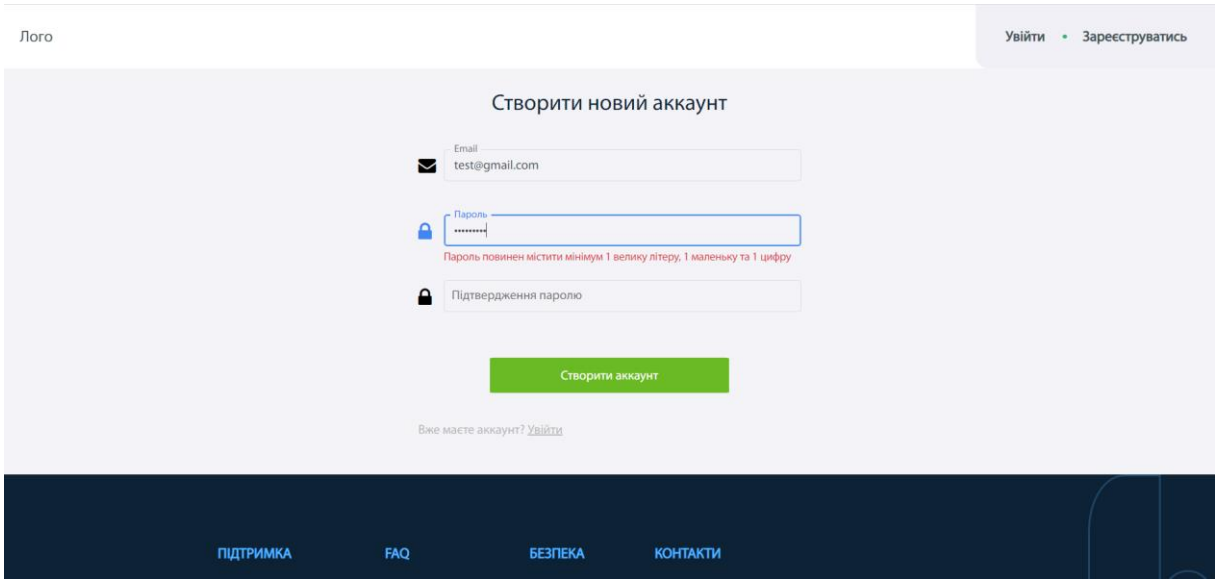


Рис. 7. Повідомлення про недостатню безпеку паролю

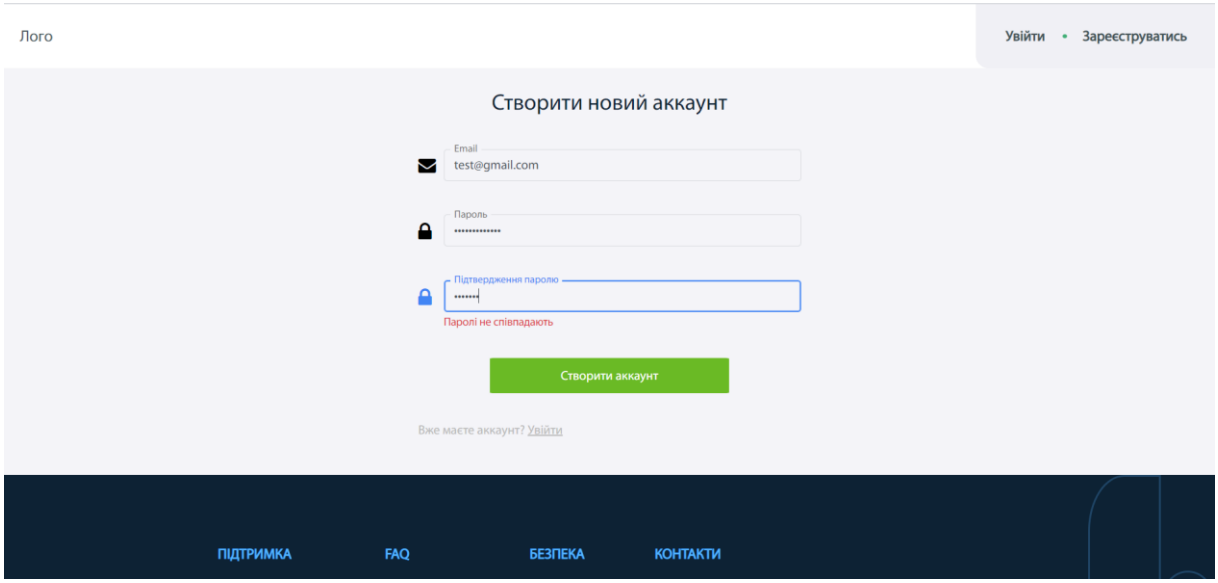


Рис. 8. Повідомлення про неспівпадіння паролів

Процедура авторизації.

При натисканні кнопки «Увійти» (рис. 1), користувач переходить до сторінки авторизації (рис. 9). Для того, щоб авторизуватись у системі користувач повинен ввести e-mail та пароль, який він вводив при реєстрації. Введення даних до всіх полів оброблено згідно необхідної валідації. Наприклад, якщо користувача з введеним e-mail не знайдено, або якщо введено неправильний пароль, то виникає помилка (рис. 10).

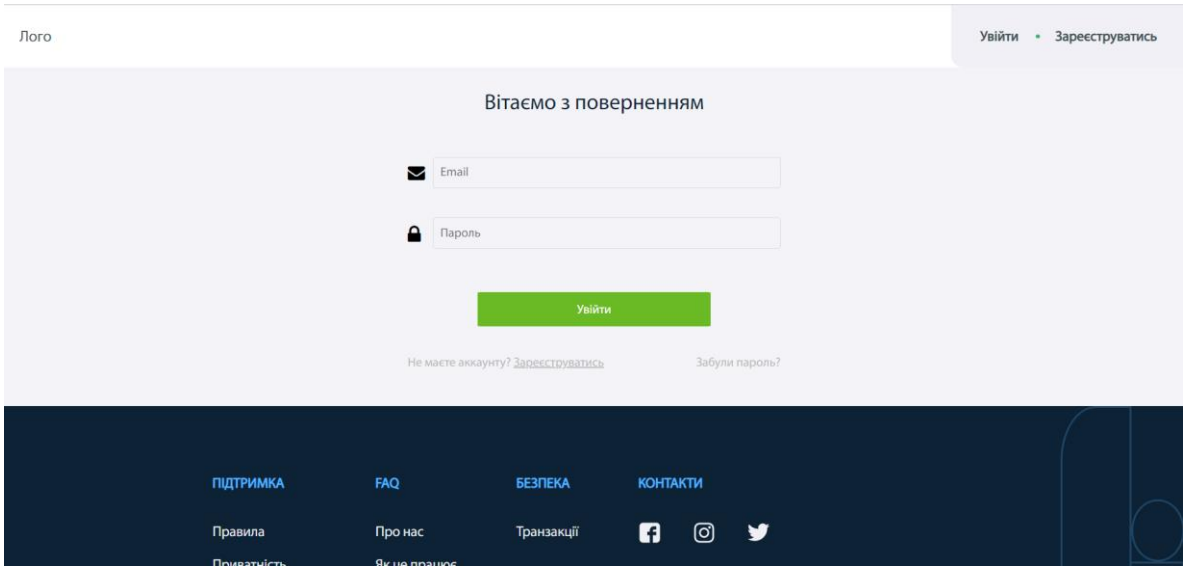


Рис. 9. Сторінка авторизації користувача

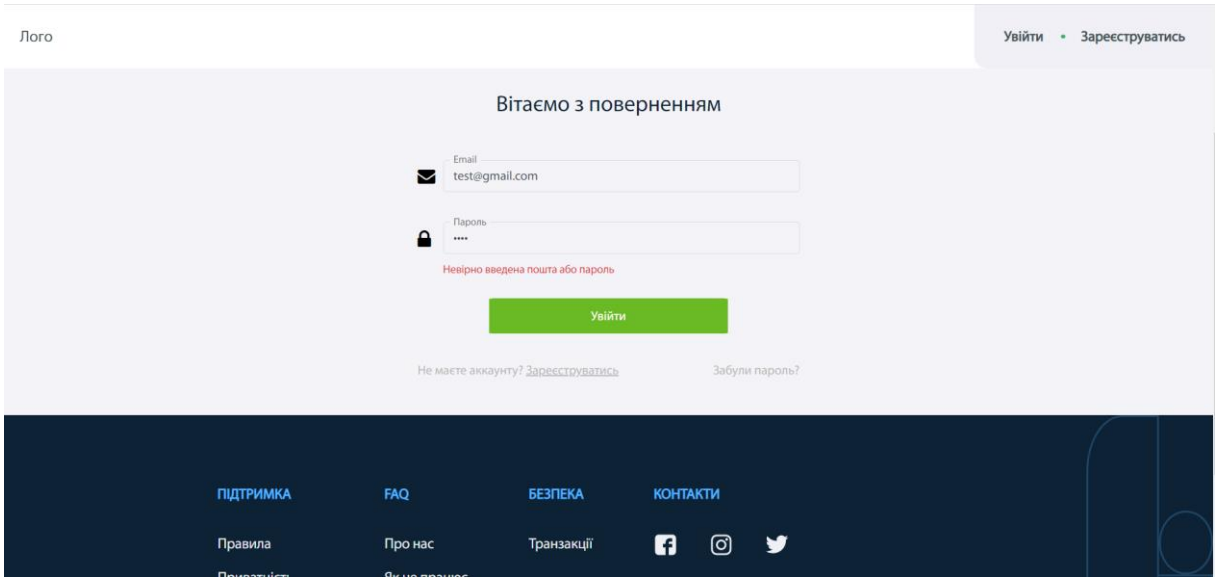


Рис. 10. Повідомлення про помилку на сторінці авторизації

4. Основні дії зареєстрованого користувача

Дії з балансами.

Для того, щоб займатись просторовим арбітражем, користувачу для початку потрібно поповнити свій баланс по будь-якій з валют. Для цього йому треба перейти до пункту «Баланси» (рис. 11).

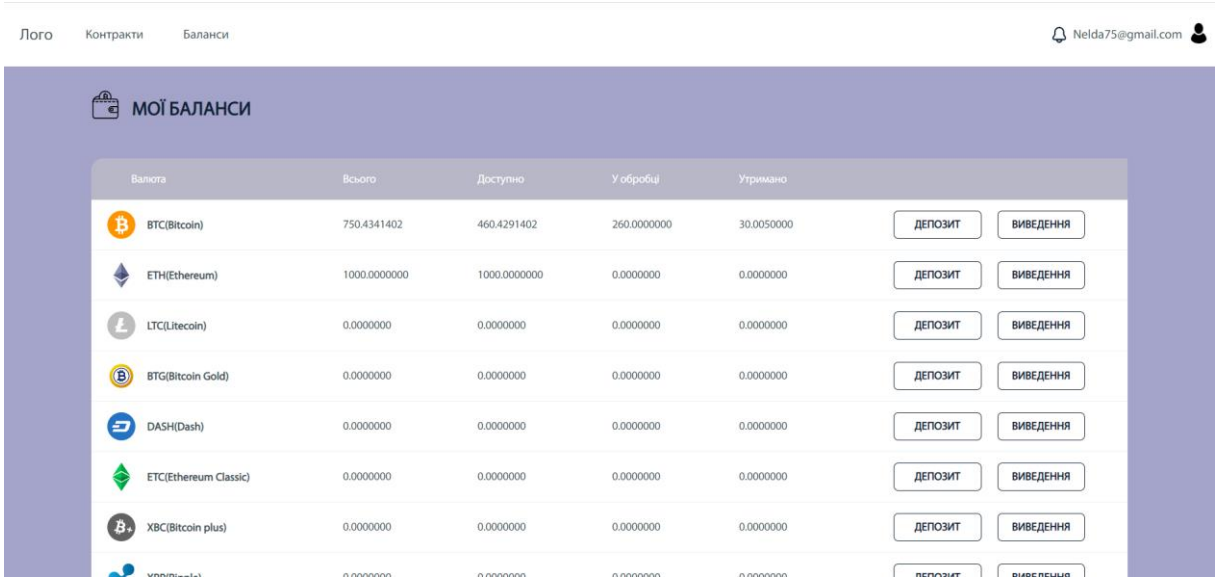


Рис. 11. Сторінка «Баланси» у додатку

На сторінці балансів (рис. 11) є наступні поля – назва валюти та її лого, скільки усього коштів у користувача по даній валюті, скільки доступно для проведення арбітражних операцій, скільки коштів у обробці по арбітражних заявках, скільки коштів утримано для виведення, та кнопки «Депозит» та «Виведення». Після натиску на кнопку «Депозит» відкривається модальне вікно (рис. 12) з інформацією про те як поповнити баланс. На даному етапі це робиться вручну, адже ми не можемо оперувати реальними грошима. При натиску на «Вивід» теж з'являється модальне вікно, де можна ввести свої реквізити, за якими користувач хоче отримати криптовалюту (рис. 13). На даний момент ця функція також виключена. Також операція виводу має валідацію введених даних, та генерує помилки. Наприклад, при невірно введеній адресі криптогаманця (рис. 14), або при перевищенні кількості доступних коштів на рахунку (рис. 15). Після підтвердження введених даних користувачу буде показано повідомлення про успішність операції (рис. 16).

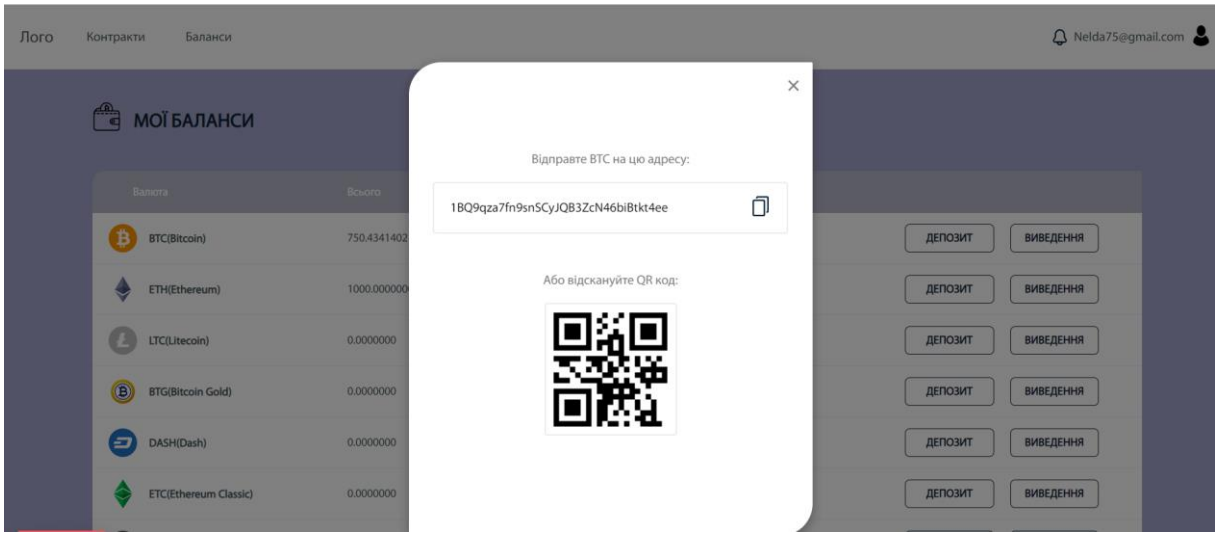


Рис. 12. Вікно депозиту коштів на баланс

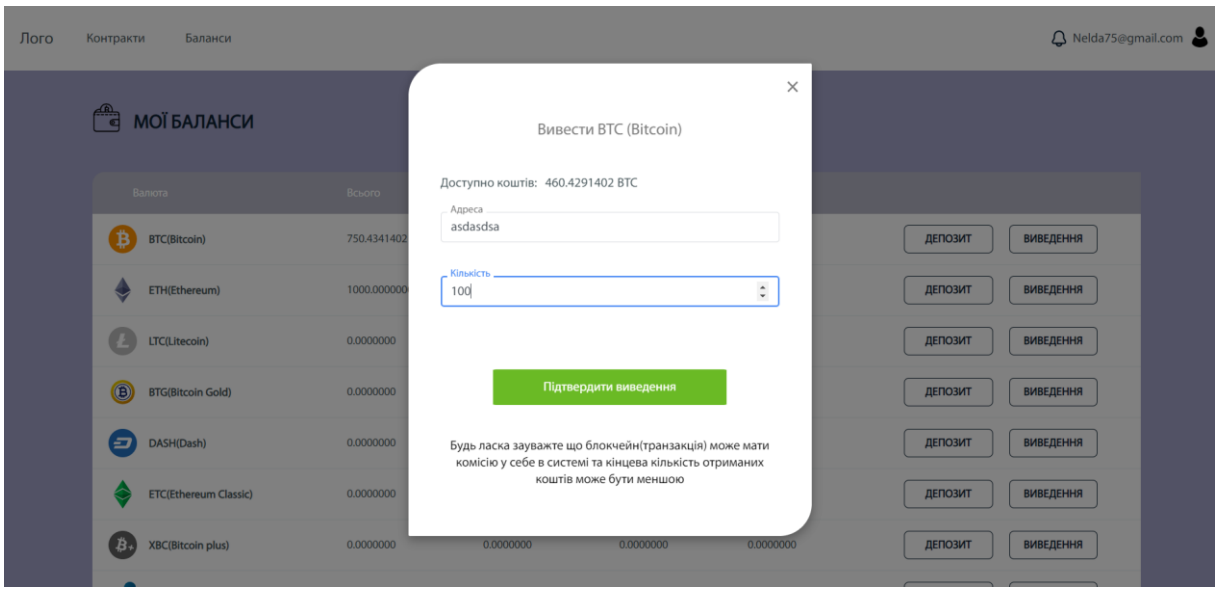


Рис. 13. Вікно виводеняя коштів з балансу

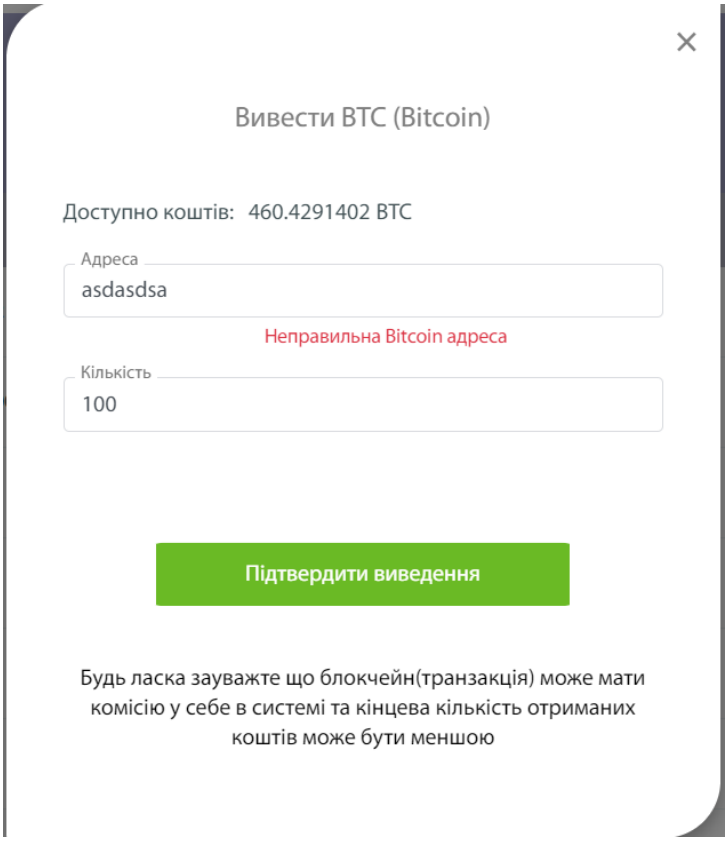


Рис. 14. Помилка при невірній адресі криптогаманця

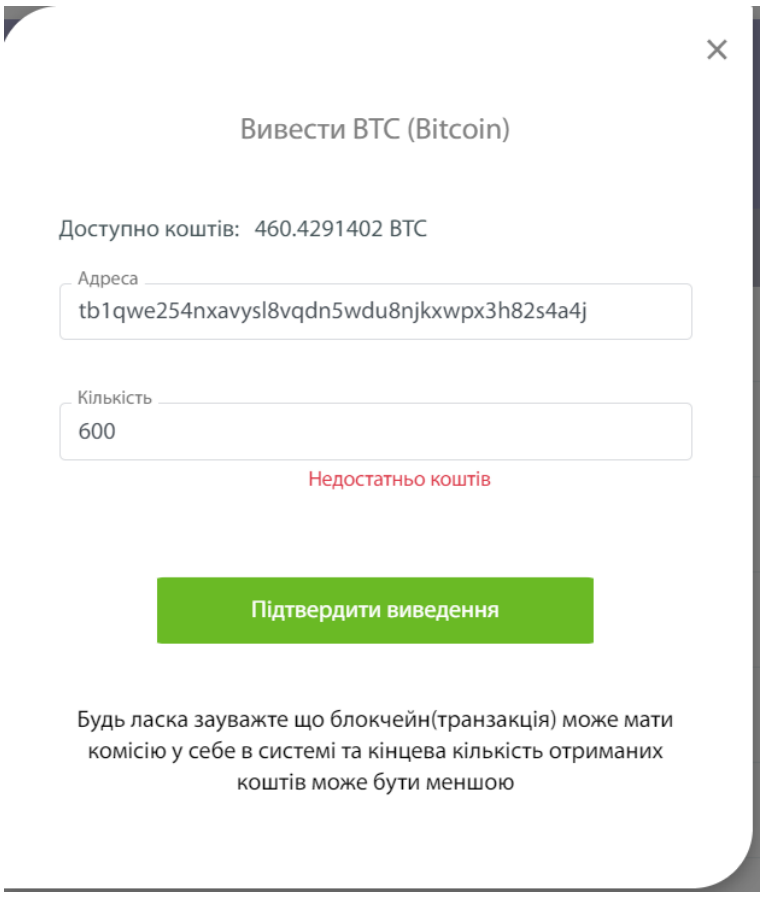


Рис. 15. Помилка при перевищенні доступних коштів

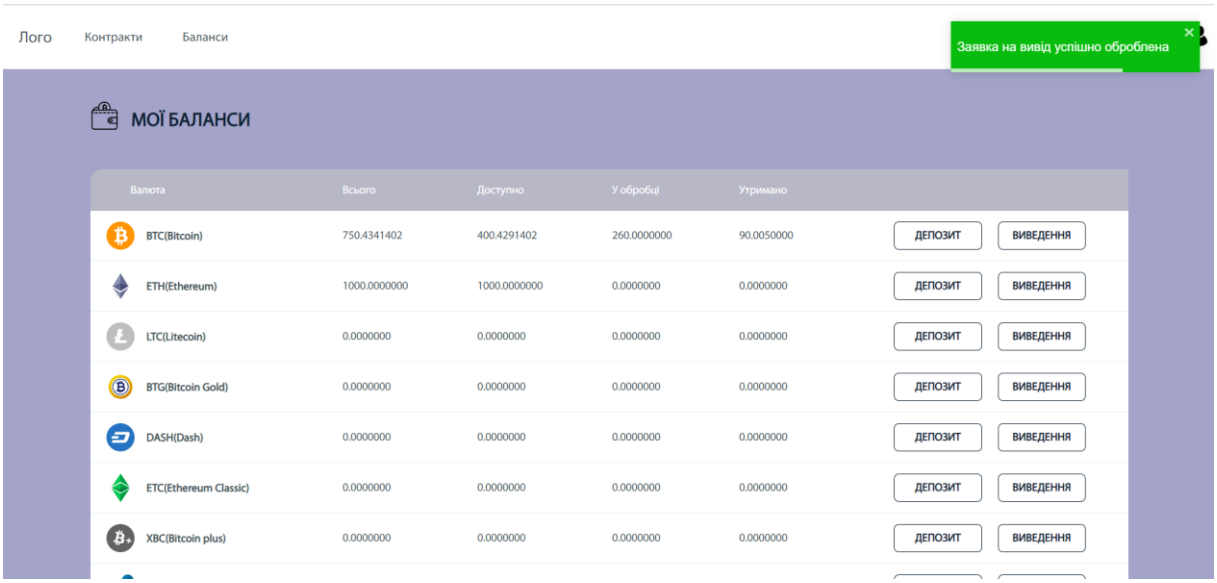


Рис. 16. Сповіщення про успішне виведення коштів

Сторінка контрактів.

Для того, щоб почати створювати арбітражні заявки користувачу спочатку потрібно перейти на сторінки «Контракти» (рис. 17). На ній

містяться доступні для торгівлі контракти, кількість доступних бірж по кожному з них, доступні кошти, та максимальний відсоток за останню добу.

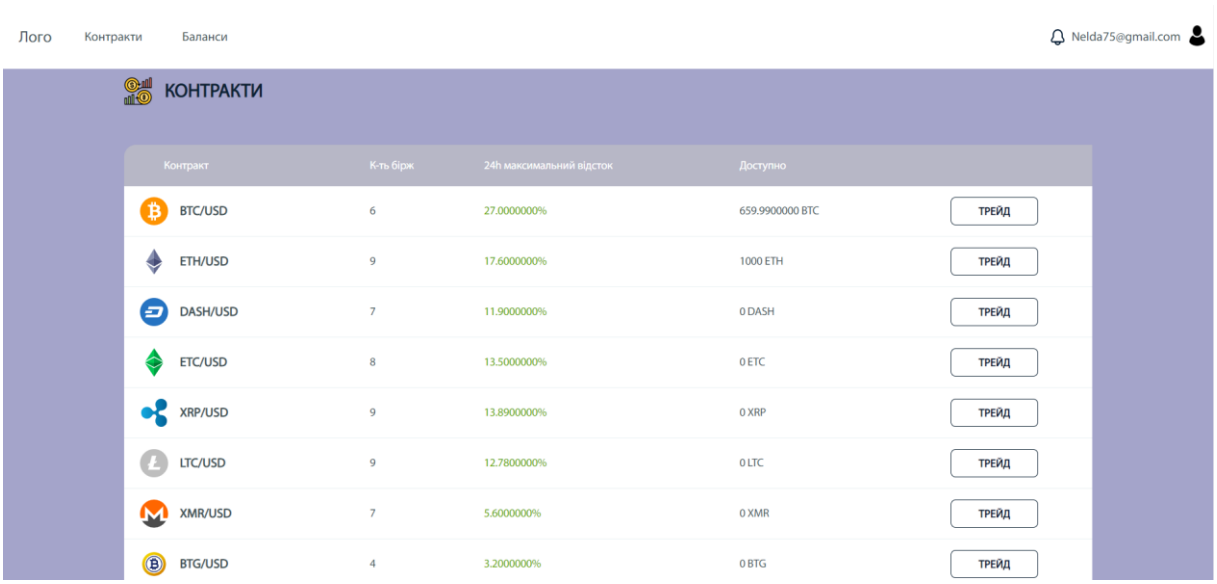


Рис. 17. Сторінка «Контракти»

5. Процедура проведення просторового арбітражу

Після натиску на кнопку «Трейд» по будь-якому з контрактів (рис. 17), користувач потрапляє на сторінку трейдингу (рис. 18).

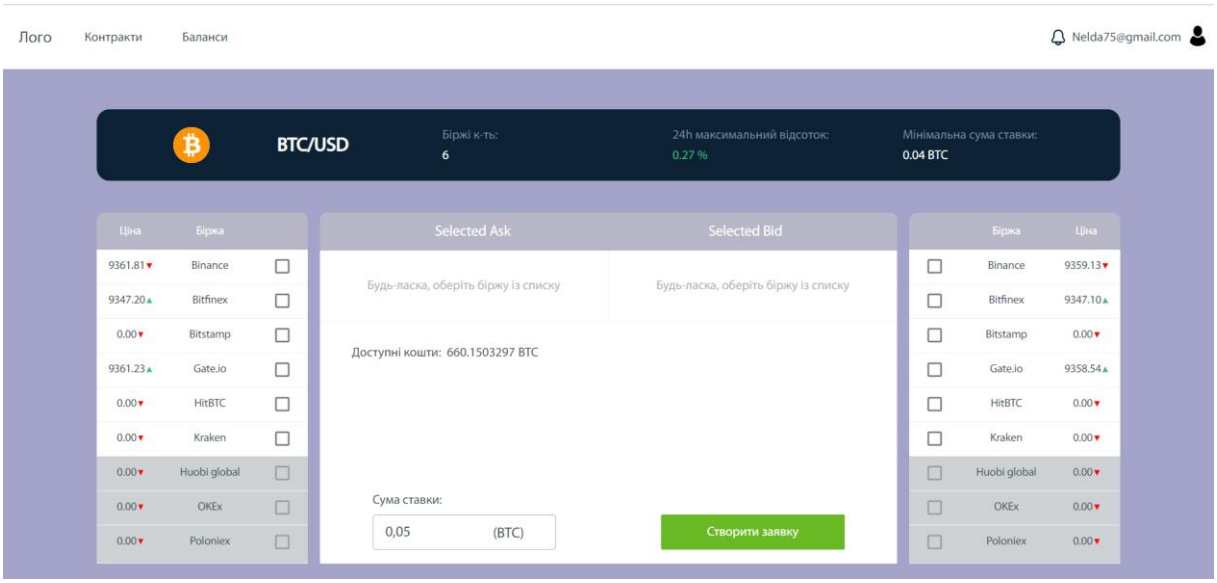


Рис. 18. Сторінка трейдингу

Для того, щоб створити арбітражну заявку користувач для початку потрібен обрати біржу для «купівлі» криптовалюти зі списку зліва, та біржу «продажу» зі списку справа, а також вказати суму, за якою він хоче створити заявку (рис. 19). Після цього заявку можна підтвердити або ж відмінити (рис. 20). Після підтвердження користувач отримає повідомлення про успішно створену заявку та її id (рис. 21).

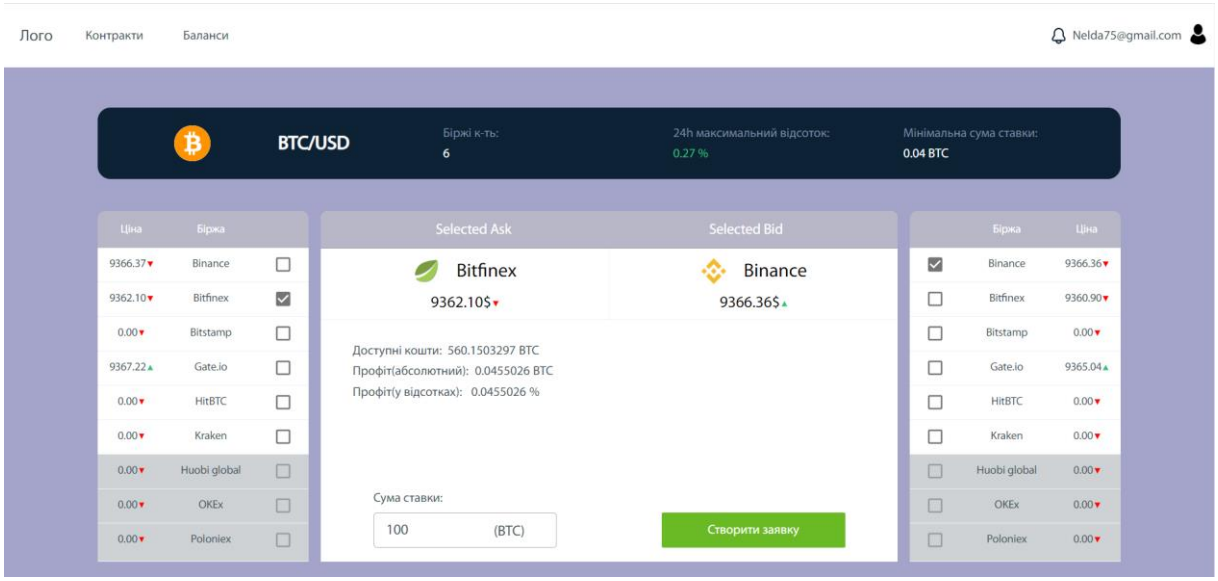


Рис. 19. Створення нової арбітражної заявки

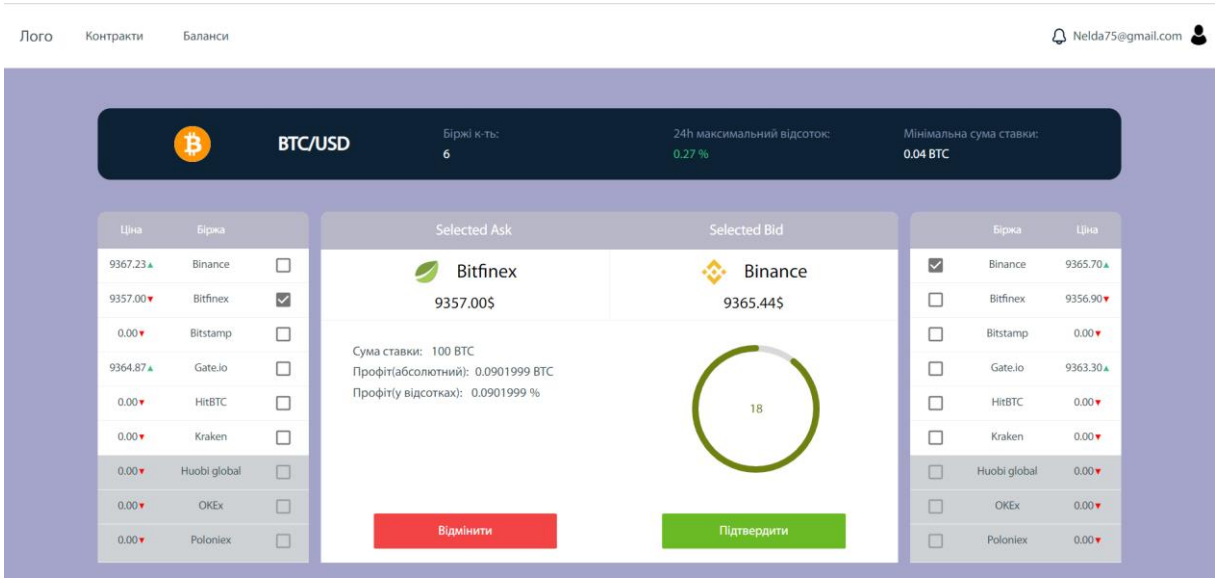


Рис. 20. Підтвердження або відміна заявки

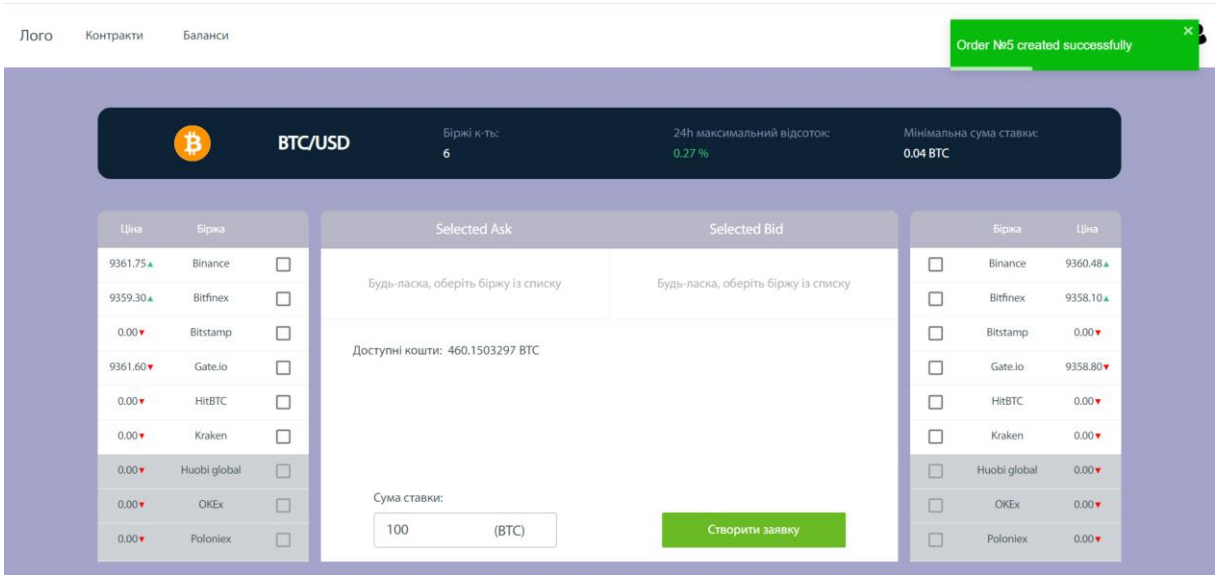


Рис. 21. Повідомлення про успішно створену заявку

Після цього заявка з’явиться в списку заявок у обробці (рис. 22). З реальними криптовалютами такі операції зазвичай займають певний час, у додатку ж створена симуляція цього. Для того, щоб заявка змінила статус на «Підтверджено» потрібно зачекати 3 хвилини (рис. 23). Після цього, щоб вона повністю обробилася потрібно зачекати ще 10 хвилин, і вона з’явиться у списку завершених заявок (рис. 24).

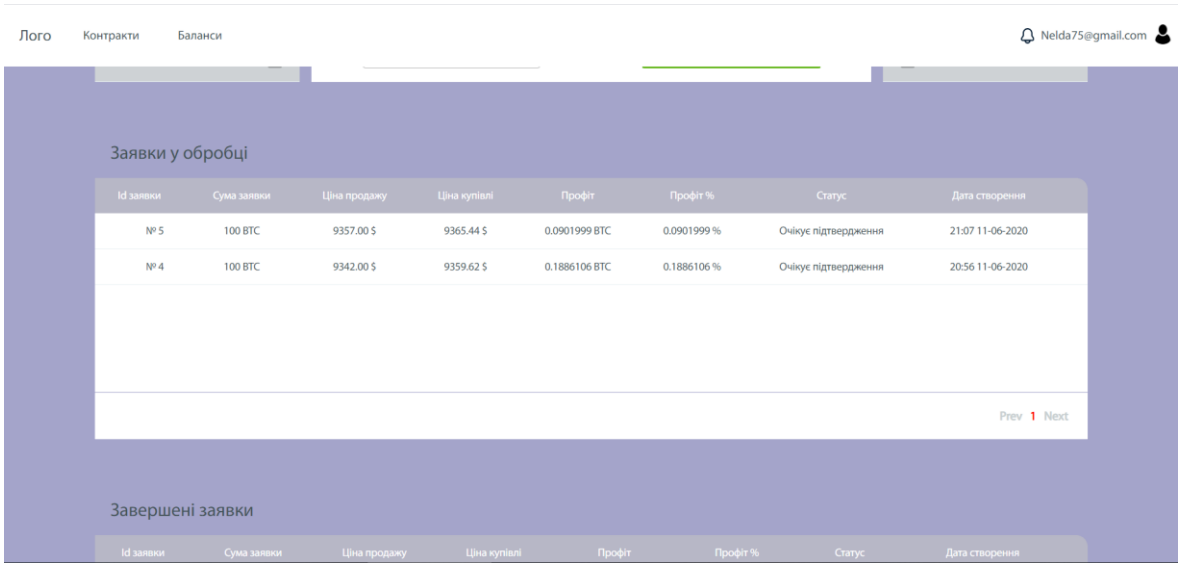


Рис. 22. Заявки у обробці очікують підтвердження

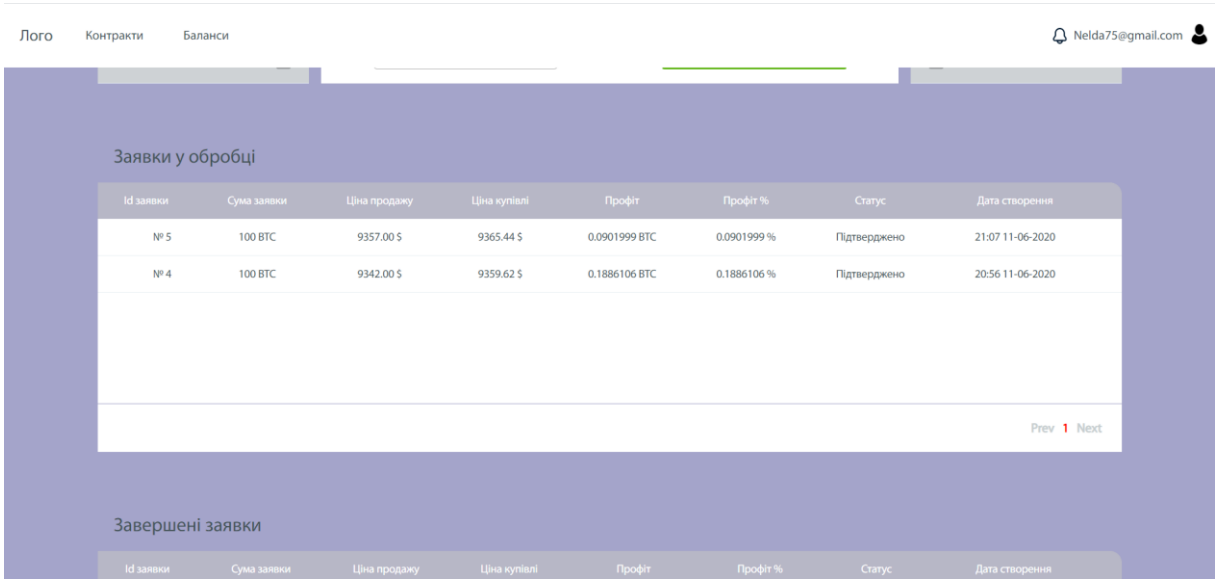


Рис. 23. Підтверджені заявки у обробці

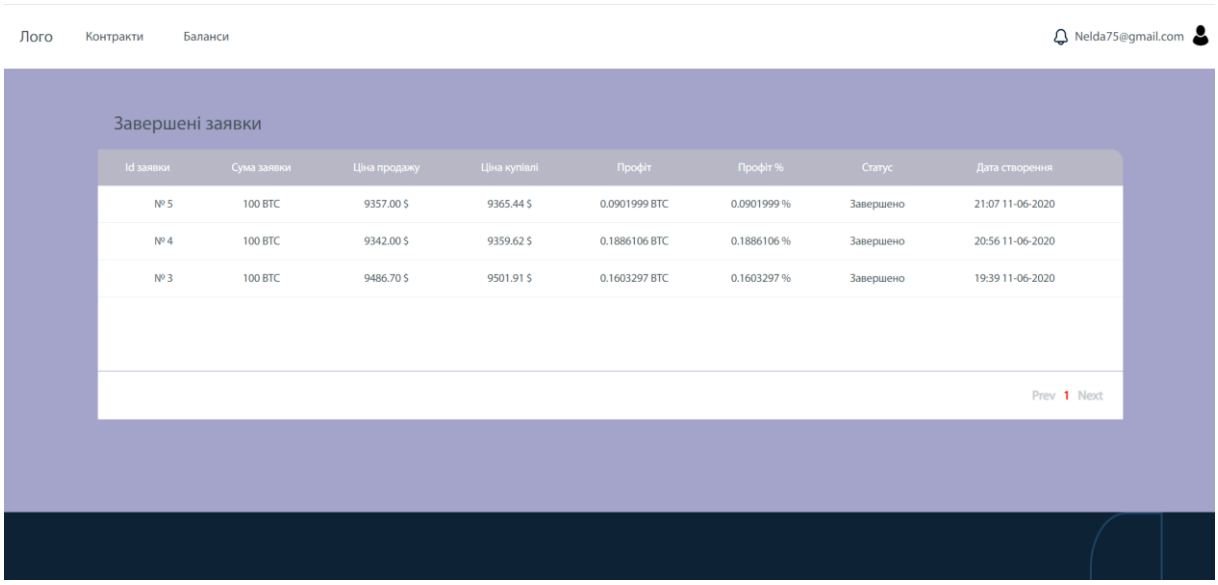


Рис. 24. Завершені заявки

6. Служба підтримки та можливості адміністратора ресурсу

Служба підтримки

Натиснувши на кнопку «Підтримка» у меню додатка (рис. 25), користувач потрапляє на сторінку звернень (рис. 26).

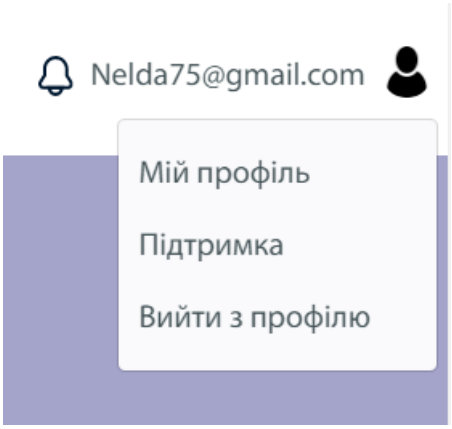


Рис. 25. Меню додатка

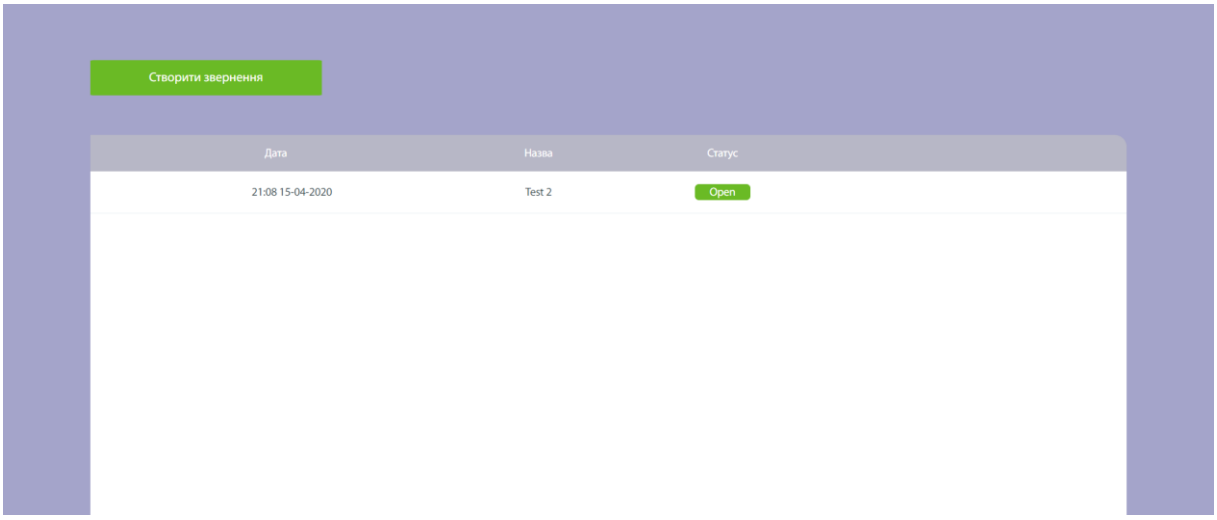


Рис. 26. Сторінка звернень

Після цього користувач може створити нове звернення, або ж перейти у будь-яке яке ще має незавершений статус. При створенні звернення користувач має ввести його назву та опис (рис. 26). Після цього воно з’явиться в списку у користувача (рис. 27), та у адміністратора ресурсу (рис. 28).

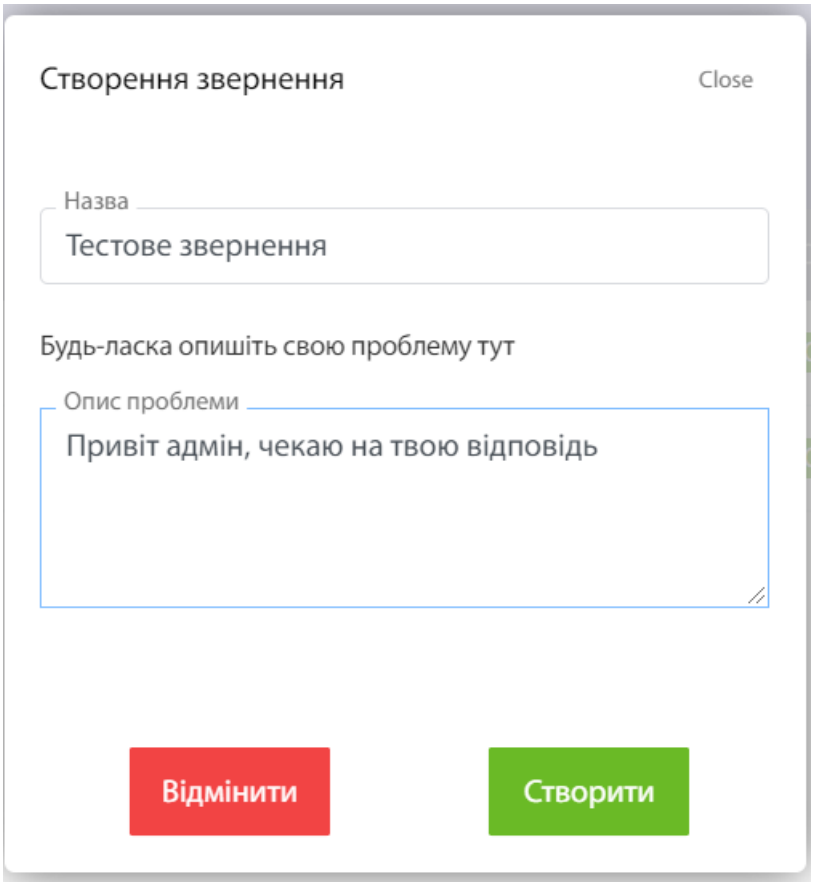


Рис. 26. Створення нового звернення

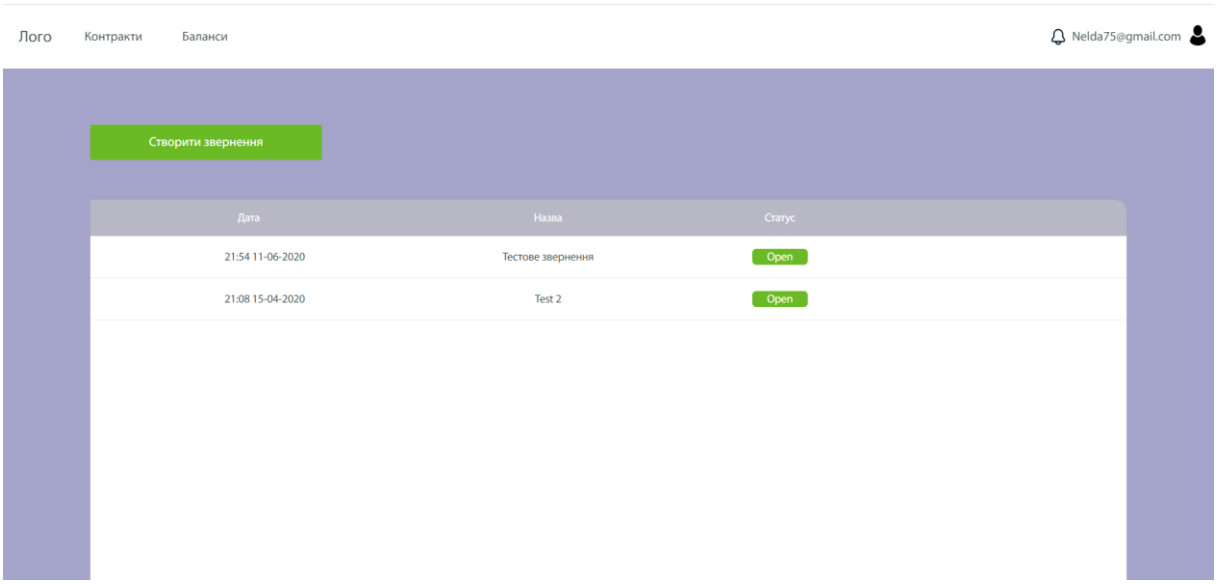


Рис. 27. Звернення в списку у користувача

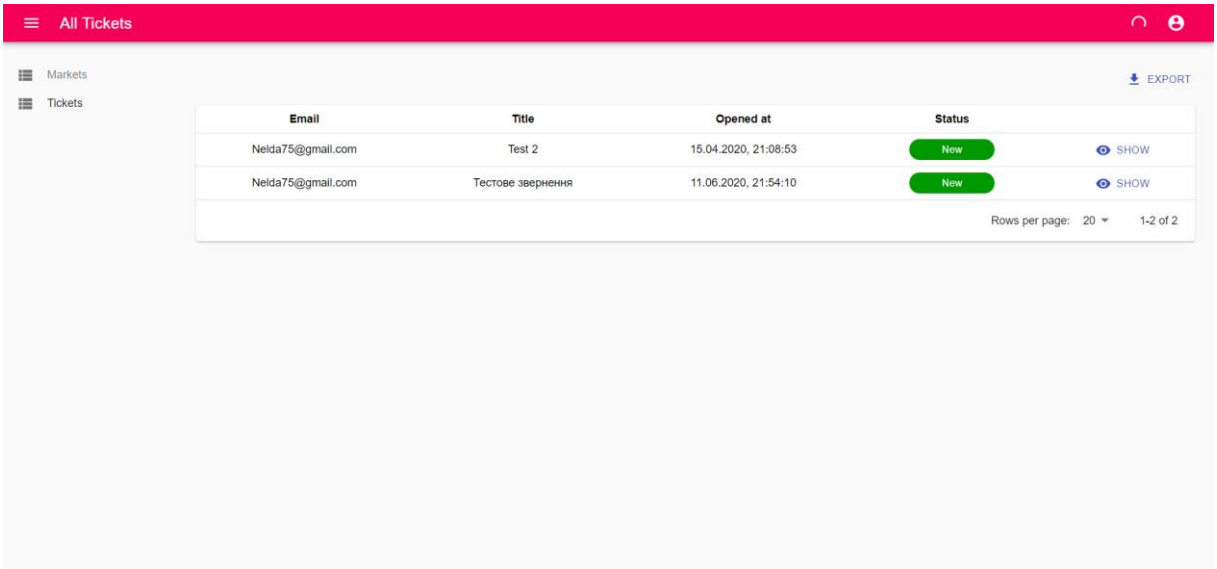


Рис. 28. Звернення у списку адміністратора

Після цього адміністратор та користувач можуть спілкуватись у реальному часі. Також після відповіді адміністратора (рис. 29), у користувача з’явиться значок нового повідомлення навпроти звернення (рис. 30).

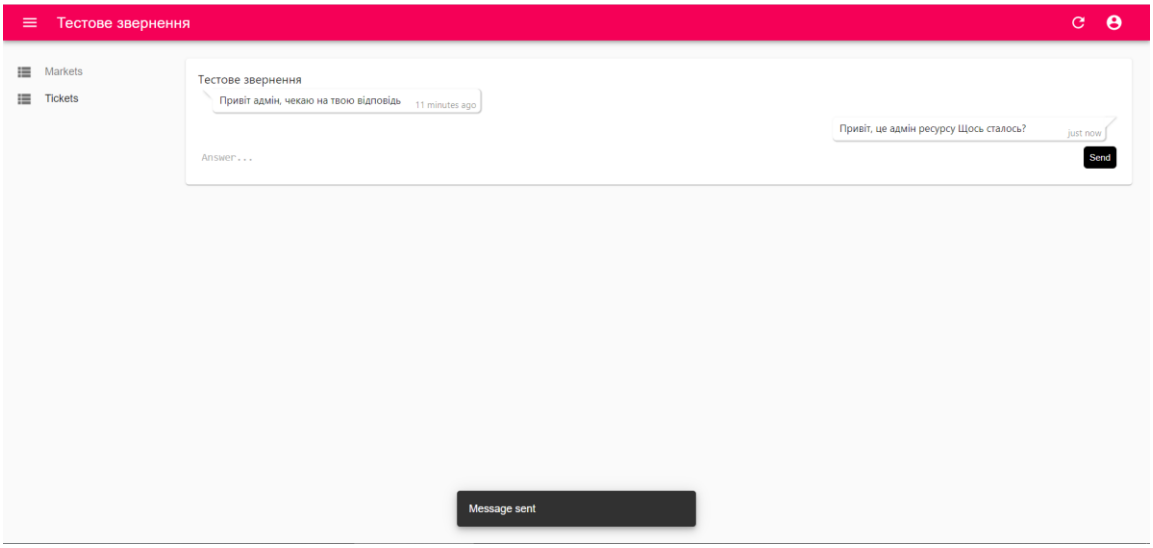


Рис. 29. Відповідь адміністратора користувачу

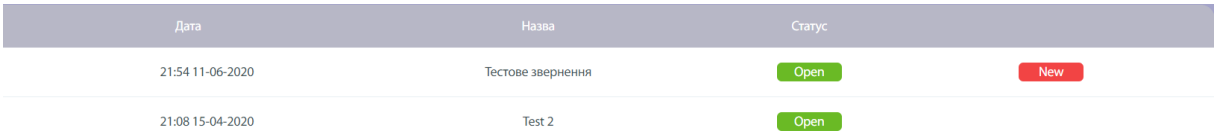


Рис. 30. Нове повідомлення у зверненні

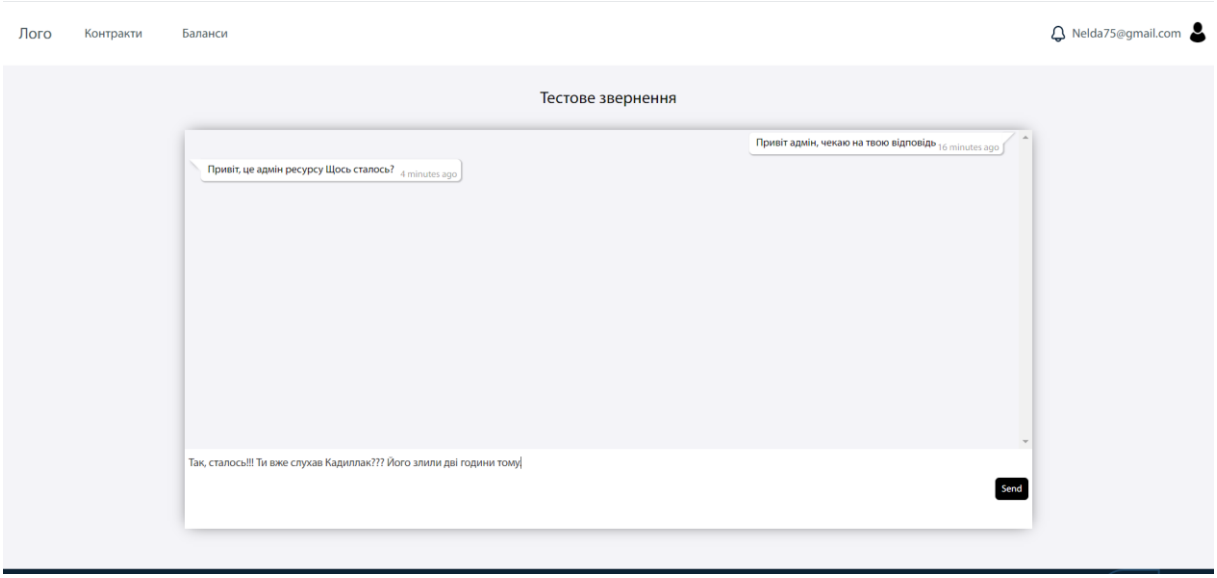


Рис. 31. Остаточний вигляд чату з адміністратором

Додаткові можливості адмінітсратора

Адміністратор має у своєму розпорядженні сторінку з контрактами, де він може відключати біржі для обраного контракту (рис. 32, 33), або ж повністю вимикати контракт від проведення по ньому будь-яких операцій (рис. 34, 35). Це створено для регулювання ситуацій коли по якійсь біржі починають проводитись підозрілі операції.

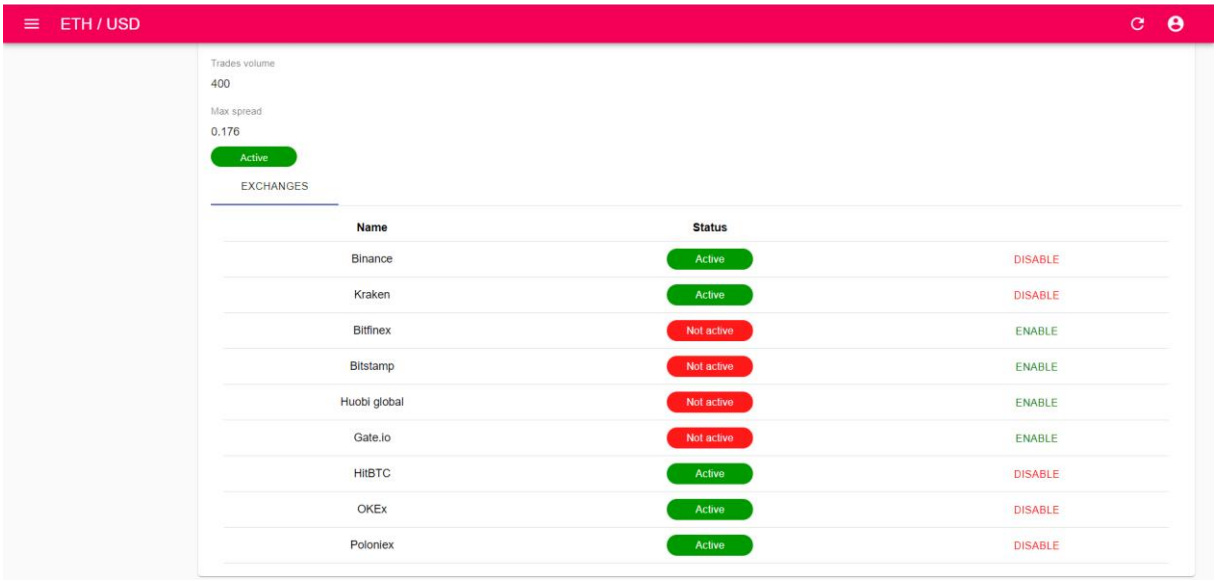


Рис. 32. Відключення бірж адміністратором

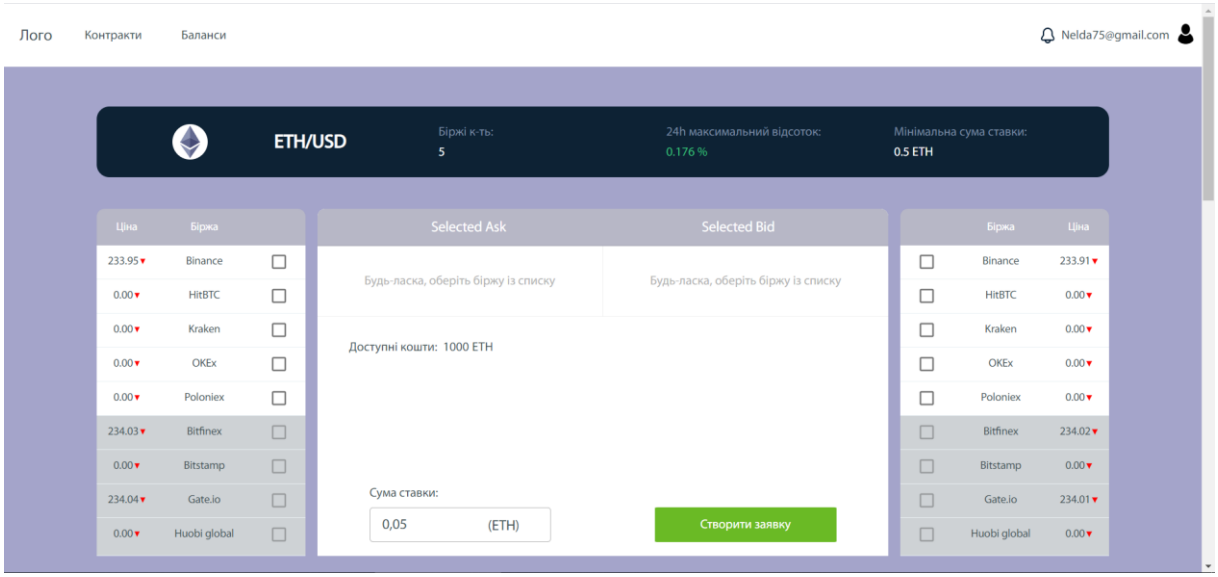


Рис. 33. Відключені біржі у користувача

Рис. 34. Відключення всього контракту адміністратором

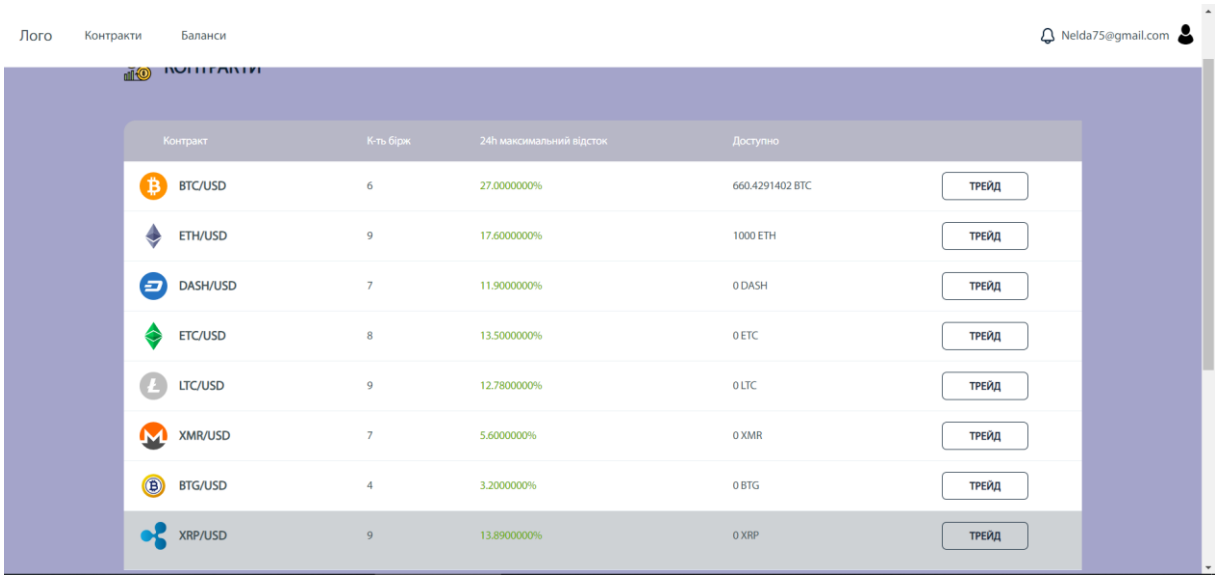
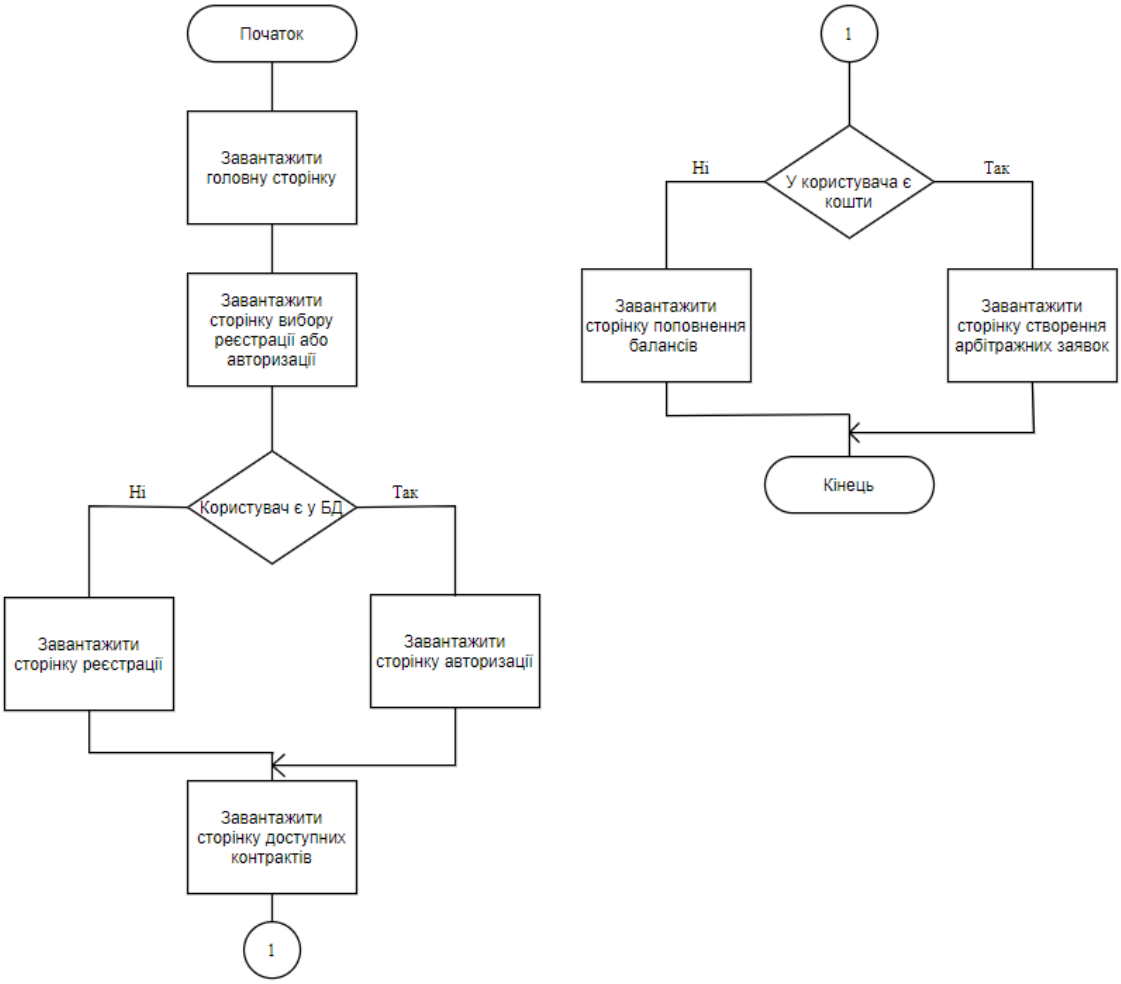


Рис. 35. Вигляд відключеної біржі у користувача

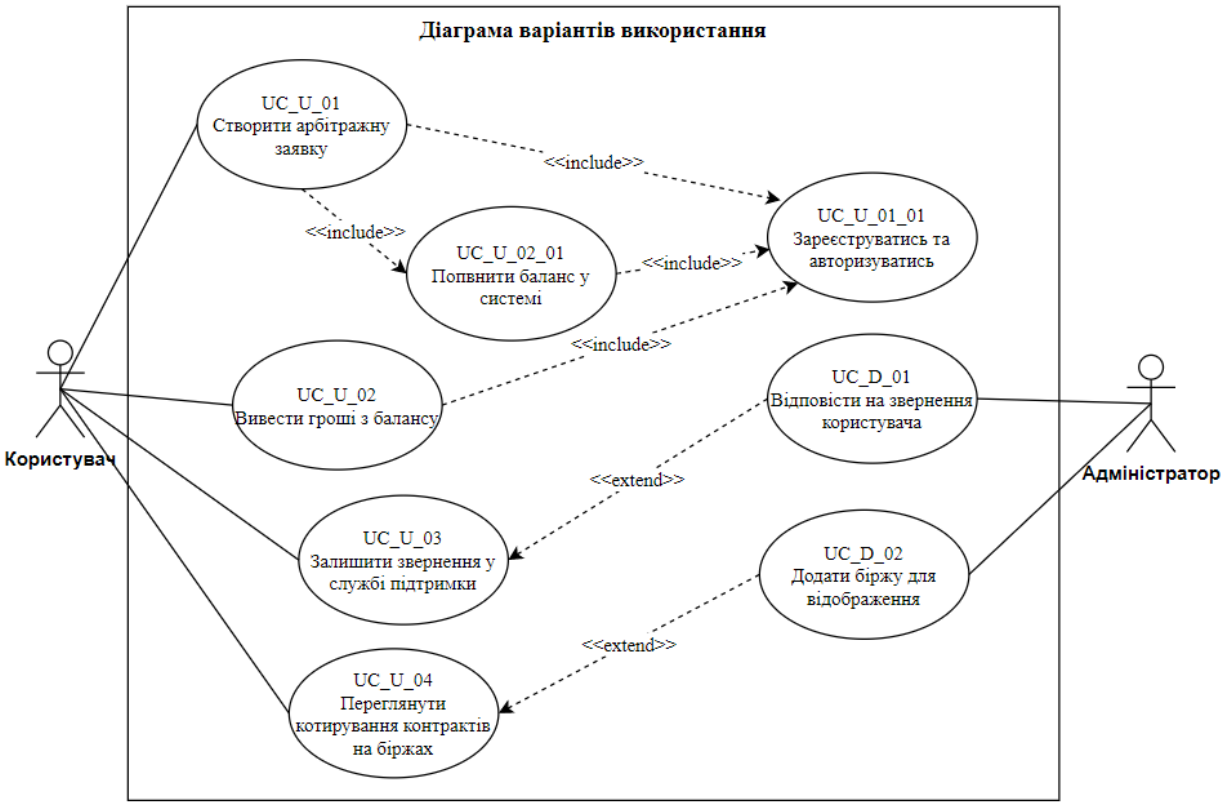
Додаток 1

Копії графічних матеріалів



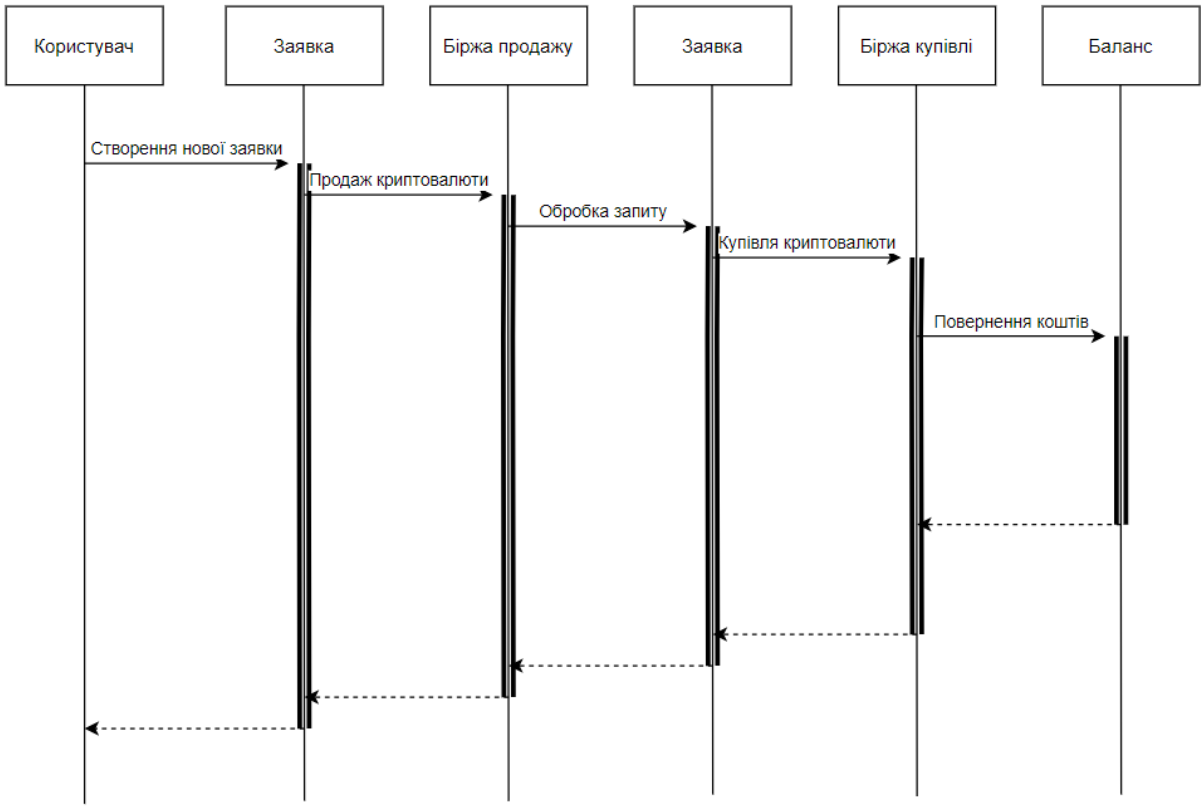
ДП.045440-06-99
Веб-додаток для просторового арбітражу криптовалют. Алгоритм взаємодії графічних елементів. Схема алгоритму

ДП.045440-07-99
Веб-додаток для просторового арбітражу криптовалютами. База даних системи. Схема даних



Андрієвський Дмитрій, група КП-61

Діаграма послідовності проведення арбітражу



Андрієвський Дмитрій, група КП-61

Додаток 2

Лістинги програм

```
import 'bootstrap-css-only/css/bootstrap.min.css';
import 'mdbreact/dist/css/mdb.css';
import React, { Component } from 'react';
import Link from 'next/link';

import {
  MDBNavbar,
  MDBNavbarBrand,
  MDBNavbarNav,
  MDBNavItem,
  MDBCollapse,
  MDBDropdown,
  MDBDropdownToggle,
  MDBDropdownMenu,
  MDBDropdownItem,
  MDBWaves
} from 'mdbreact';

const AuthHeader = props => {
  const [cursorPos, setCursorPos] = React.useState({});
  const handleClick = e => {
    e.stopPropagation();
    // Waves - Get Cursor Position
    const cursorPosition = {
      top: e.clientY,
      left: e.clientX,
      time: Date.now() // time indicates particular clicks
    };
    console.log(cursorPosition);
    setCursorPos(cursorPosition);
  };

  return (
    <MDBNavbarNav className="left-nav" left>
      <MDBNavItem onClick={props.toggleCollapse('navbarCollapse3')}>
        <Link href="/markets">
          <div
            className="Ripple-parent"
```

```
            onTouchStart={handleClick}
            onMouseUp={handleClick}
          >
            <a>Контракти</a>
            <MDBWaves cursorPos={cursorPos} />
          </div>
        </Link>
      </MDBNavItem>
      <MDBNavItem onClick={props.toggleCollapse('navbarCollapse3')}>
        <Link href="/balances">
          <div
            className="Ripple-parent"
            onTouchStart={handleClick}
            onMouseUp={handleClick}
          >
            <a>Баланси</a>
            <MDBWaves cursorPos={cursorPos} />
          </div>
        </Link>
      </MDBNavItem>
    </MDBNavbarNav>
  );
};

class NavbarPage extends Component {
  constructor(props) {
    super(props);
    this.state = {
      collapseID: ''
    };
  }

  toggleCollapse = collapseID => () =>
    this.setState(prevState => ({
      collapseID: prevState.collapseID !== collapseID ? collapseID : null
    }));

  render() {
```

```
const { user, logout } = this.props;
const { collapseID } = this.state;

return (
  <div className="app-header">
    <MDBNavbar
      expand="md"
      fixed="top"
      scrolling
      scrollingNavbarOffset={250}
    >
      <MDBNavbarBrand>
        <Link href="/">
          <strong>Logo</strong>
        </Link>
      </MDBNavbarBrand>
      <div>
        <img
          src="/static/images/menu-button.svg"
          className="hamburger"
          onClick={this.toggleCollapse('navbarCollapse3')}
        />
      </div>
      <MDBCollapse id="navbarCollapse3" isOpen={collapseID} navbar>
        {user && <AuthHeader toggleCollapse={this.toggleCollapse} />}
        {!user ? (
          <div className="reg-info">
            <div onClick={this.toggleCollapse('navbarCollapse3')}>
              <Link href="/login">
                <a>Увійти</a>
              </Link>
            </div>
            <div className="circle-container">
              <div className="circle" />
            </div>
            <div onClick={this.toggleCollapse('navbarCollapse3')}>
              <Link href="/signup">
                <a>Зареєструватись</a>
```

```
                  </Link>
                </div>
              </div>
            ) : (
              <MDBNavbarNav right className="userInfo">
                <MDBNavItem>
                  <a
                    style={{ marginRight: '8px' }}
                    className="waves-effect waves-light d-flex align-items-
center"
                    to="#!"
                  >
                    {user.notification ? (
                      <img
                        src="/static/images/notification.svg"
                        style={{
                          height: '8px',
                          position: 'absolute',
                          top: '0px',
                          left: '10px'
                        }}
                      />
                    ) : (
                      <></>
                    )}
                    <img
                      src="/static/images/bell.svg"
                      style={{ height: '20px' }}
                    />
                  </a>
                </MDBNavItem>
                <MDBNavItem>
                  <p style={{ margin: '0px', fontWeight: 500
}}>{user.email}</p>
                </MDBNavItem>

                <MDBDropdown>
                  <MDBDropdownToggle className="dopdown-toggle" nav>
```

```
                    <img
                      src="/static/images/icons/profile.png"
                      className="rounded-circle z-depth-0"
                      style={{ height: '25px', padding: 0 }}
                      alt=""
                    />
                  </MDBDropdownToggle>
                  <MDBDropdownMenu className="dropdown-default" right>
                    <MDBDropdownItem href="#!">Мій профіль</MDBDropdownItem>
                    <MDBDropdownItem
href="/tickets">Support</MDBDropdownItem>
                    <MDBDropdownItem
                      href="#!"
                      onClick={() => {
                        this.toggleCollapse('navbarCollapse3')();
                        logout();
                      }}
                    >
                      Вийти
                    </MDBDropdownItem>
                  </MDBDropdownMenu>
                </MDBDropdown>
              </MDBNavbarNav>
            )}
          </MDBCollapse>
        </MDBNavbar>
      </div>
    );
  }
}

export default NavbarPage;
```

```
import React from 'react';
import { MDBInput } from 'mdbreact';
import Router from 'next/router';
import Button from './buttons/bsButton';

const SignUp = props => {
  const {
    onSubmit,
    onEmailChange,
    onPasswordChange,
    onConfirmPasswordChange,
    validate,
    resetFieldError,
    loading,
    errors,
    serverError
  } = props;
  return (
    <div className="signup-form-container">
      <div className="main-icon-wrapper">
        <img src="/static/images/icons/bitcoin-safety-lock-symbol.svg" />
      </div>
      <h1>Create new account</h1>
      <div className="signup-form">
        <div className="field-container">
          <MDBInput
            label="Email"
            icon="envelope"
            outline
            onChange={onEmailChange}
            onBlur={() => validate('email')}
            onFocus={() => resetFieldError('email')}
          />
          <div className="error-feedback feedback-large">
            <span>{errors.email}</span>
          </div>
        </div>
        <div className="field-container">
```

```
            <MDBInput
              label="Пароль"
              icon="lock"
              type="password"
              outline
              onChange={onPasswordChange}
              onBlur={() => validate('password')}
              onFocus={() => resetFieldError('password')}
            />
            <div className="error-feedback feedback-large">
              <span>{errors.password}</span>
            </div>
          </div>
          <div className="field-container">
            <MDBInput
              label="Підтвердження паролю"
              icon="lock"
              type="password"
              outline
              onChange={onConfirmPasswordChange}
              onBlur={() => validate('confirmPassword')}
              onFocus={() => resetFieldError('confirmPassword')}
            />
            <div className="error-feedback feedback-large">
              <span>{errors.confirmPassword || serverError}</span>
            </div>
          </div>
          <Button value="Створити аккаунт" onClick={onSubmit} loading={loading}
/>
        </div>
        <div className="links">
          <div role="button" tabIndex={0} onClick={() =>
Router.push('/login')}>
            <h4>
              Вже маєте аккаунт?
              <span> </span>
              <u>Log in</u>
            </h4>
```

```
        </div>
        {/* <div role="button" tabIndex={0} onClick={() =>
Router.push('/signup')}>
          <h4>Забули пароль?</h4>
        </div> */}
      </div>
    </div>
  );
};
export default SignUp;
```

Додаток 3

Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ

ВЕБ-ДОДАТОК ДЛЯ ПРОСТОРОВОГО АРБІТРАЖУ КРИПТОВАЛЮТ

Виконав: Андрієвський Дмитрій Олександрович

Керівник: Старший викладач кафедри ПЗКС, к.т.н., Хіцко Я.В.

Київ – 2020

ПОСТАНОВКА ЗАДАЧІ

Мета проекту: розробити веб-додаток для просторового арбітражу криптовалют.

Завдання:

1. Проаналізувати предметну область, визначити її актуальність та проблематику, а також існуючі програмні рішення. Визначити вимоги до розроблення програмного додатка.
2. Розробити програмний додаток згідно вимог за допомогою обраних засобів реалізації.
3. Протестувати розроблений програмний додаток та проаналізувати його відповідність до вимог.

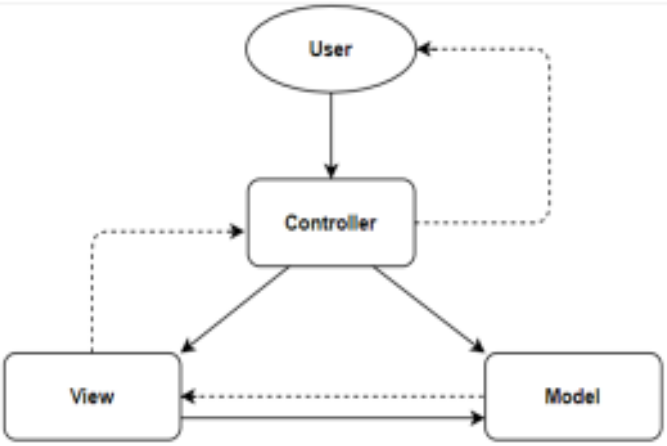
АКТУАЛЬНІСТЬ

Арбітраж на біржах існував завжди, та з появою криптовалют отримав новий виток розвитку. Не дивлячись на те що криптовалюти потроху втрачають свою популярність, бірж трейдингу ними стає все більше. Через те що криптовалютний арбітраж з'явився відносно недавно, інструментів для його проведення все ще небагато, тому є потреба у його створенні. Такий інструмент повинен бути доступним, тому веб-додаток – найкраще рішення.

Тож, розробка програмного забезпечення у сфері інтернет-трейдингу є актуальною ідеєю.

АРХІТЕКТУРА СИСТЕМИ

Архітектура програмної системи створена на основі шаблону проектування MVC.



АРХІТЕКТУРА СИСТЕМИ

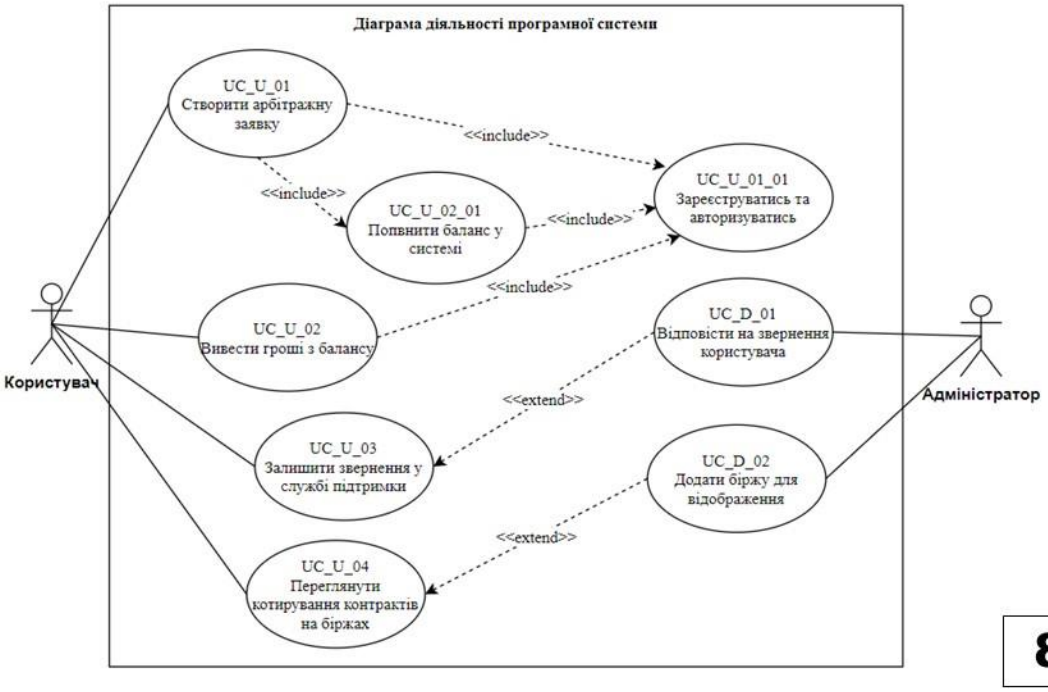
Систему моніторингу за котируваннями на біржах було створено з огляду на **мікросервісну архітектуру**.

Скрипт нагляду за кожною біржею запускається окремо. Також кожен з цих скриптів змінюється незалежно від усієї системи і може працювати зовсім по-іншому ніж інші скрипти. Управління скриптами реалізовано за допомогою інструменту RabbitMQ.

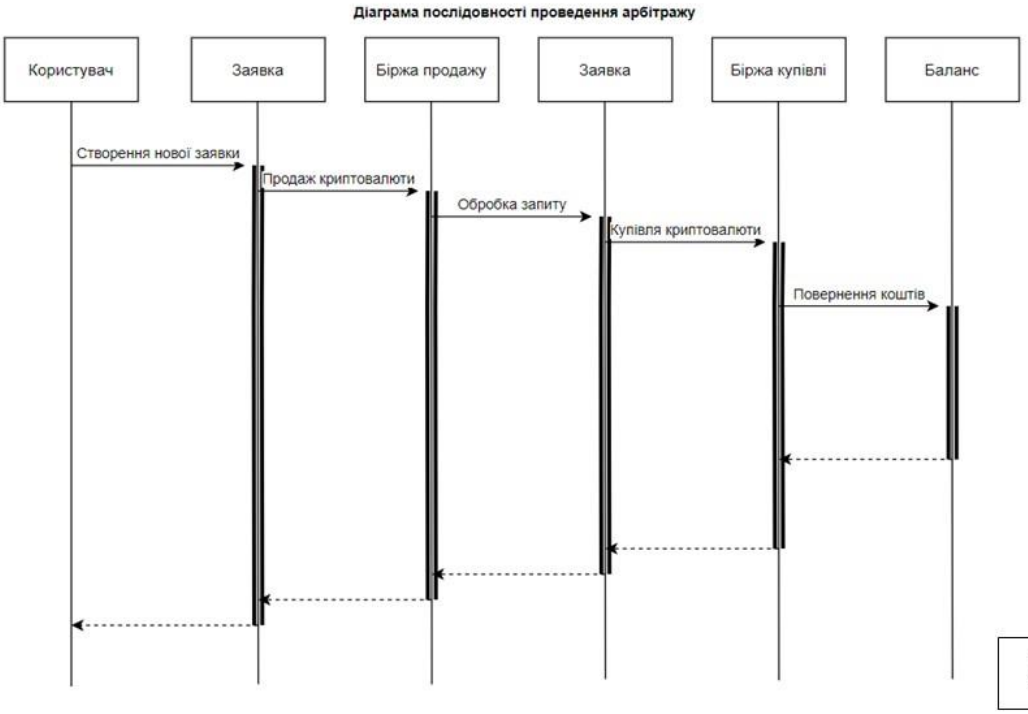
ЗАСОБИ РОЗРОБЛЕННЯ

СХЕМА БАЗИ ДАНИХ

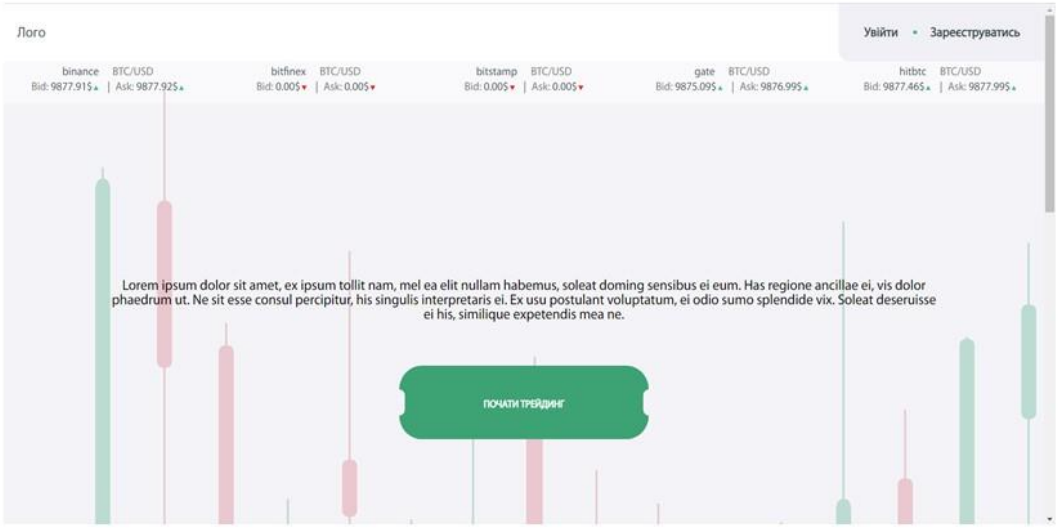
ДІАГРАМА ДІЯЛЬНОСТІ ПРОГРАМНОЇ СИСТЕМИ



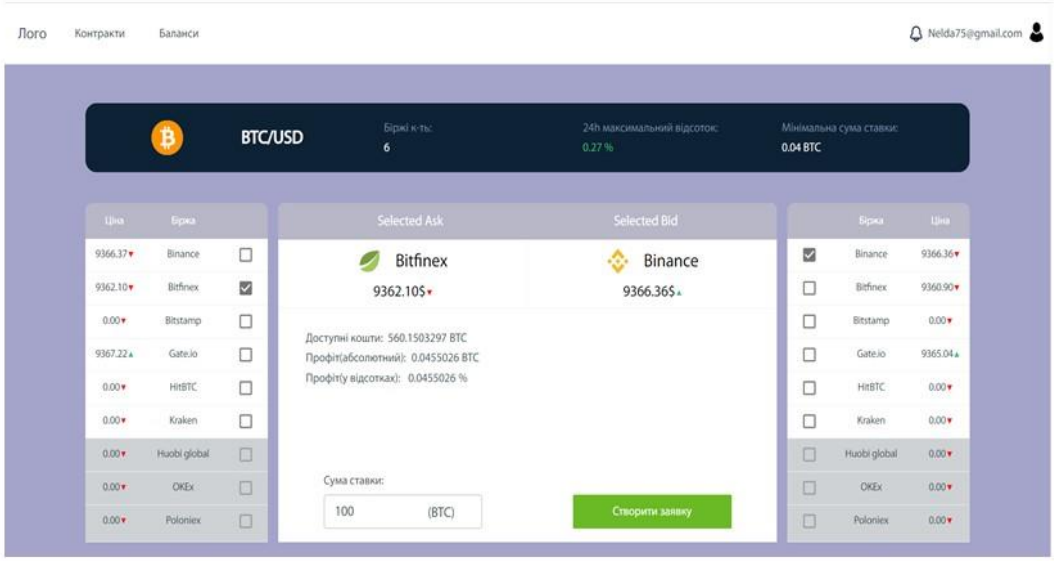
ПОСЛІДОВНІСТЬ ПРОВЕДЕННЯ ПРОСТОРОВОГО АРБІТРАЖУ



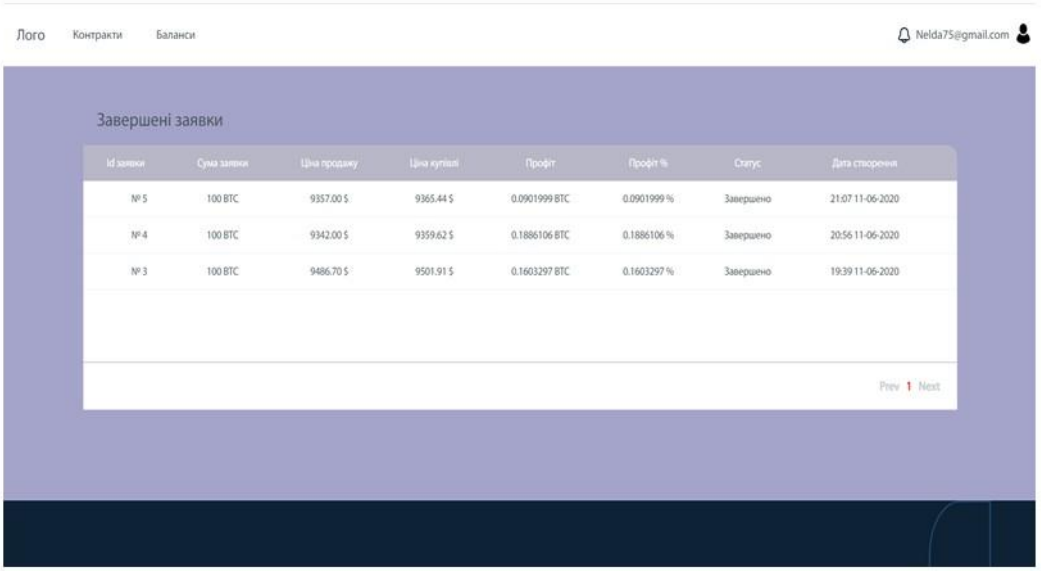
ПРИКЛАДИ ВИКОРИСТАННЯ ВЕБ-ДОДАТКА



ПРИКЛАДИ ВИКОРИСТАННЯ ВЕБ-ДОДАТКА



ПРИКЛАДИ ВИКОРИСТАННЯ ВЕБ-ДОДАТКА



КРИТЕРІЇ ОЦІНЮВАННЯ ЯКОСТІ РОЗРОБЛЕНОГО ПЗ

Відповідність розробленим вимогам до ПЗ:

- реєстрація та авторизація в системі (за допомогою e-mail, є зв'язок з електронною поштою);
- відображення котирувань криптовалют у реальному часі;
- можливість поповнення балансу та зняття коштів;
- інструмент створення арбітражних заявок;
- перегляд історії по арбітражним заявкам;
- спілкування із службою підтримки у реальному часі;

КРИТЕРІЇ ОЦІНЮВАННЯ ЯКОСТІ РОЗРОБЛЕНОГО ПЗ

Відповідність розробленим вимогам до ПЗ:

- система сповіщень користувача про дії у додатку;
- управління контрактами з панелі адміністратора;
- управління біржами з панелі адміністратора;

РЕЗУЛЬТАТИ ТЕСТУВАННЯ

У процесі тестування методом «чорного ящика» було виявлено кілька помилок у інтерфейсі та обробці даних:

- було знайдено граматичні помилки у графічних об’єктах;
- критична помилка при перевищенні доступних коштів при створенні арбітражної заявки – заявка створювалась, та баланс ставав від’ємним.

Перша помилка була зроблена через людський фактор, яку неможливо було відслідкувати під час автоматичного тестування. Друга помилка була пов’язана з нереалізованою валідацією під час створення арбітражних заявок.

ПОРІВНЯННЯ З АНАЛОГАМИ

Назва	Інтерфейс	Наявність просторового арбітражу	Використання початківцями	Безкоштовне використання	Служба підтримки
«Межбиржевой арбитраж криптовалют»	Складний	Так	Ні	Ні	Ні
«WesternPips CryptoTrader 1.7»	Складний	Ні	Ні	Ні	Ні
«MetaStock»	Складний	Так	Ні	Ні	Ні
Розроблений додаток	Простий та інтуїтивний	Так	Так	Так	Так

РЕКОМЕНДАЦІЇ ДЛЯ ПОКРАЩЕННЯ

- реалізація інструменту автоматичного створення арбітражних заявок при досягненні котируваннями заданої наперед ціни;
- додання можливості арбітражу на одній біржі між різними валютами;
- можливість реєструватись та авторизуватись за допомогою Google, Facebook;
- додання можливості пошуку та сортування бірж по статистичним даним;
- додання системи нагадувань по пошті про завершені арбітражні заявки;
- можливість залишати відгуки про біржі криптовалют

ВИСНОВКИ

1. Був проведений аналіз існуючих програмних рішень.
2. Запропоновано архітектуру ПЗ на основі шаблону проектування MVC.
3. Були визначені функціональні вимоги до розроблюваної системи, серед яких є апаратні вимоги, вимоги до інтерфейсу, вимоги до технологій та операційні вимоги. Також було визначено й нефункціональні вимоги.
4. Розроблено ПЗ, яке задовольняє критерії якості ПЗ.
5. ПЗ було протестовано та порівняно з аналогами за критеріями, які були визначені при аналізі існуючих рішень (переваги на недоліки).
6. Веб-додаток відповідно до визначених вимог. Тестування веб-додатка виконано в повному обсязі згідно наявної методики тестування.
7. Розроблений веб-додаток допомагає користувачам економити свій час, автоматизувати прості процеси та уникнути помилок пов'язаних через людський фактор.

Дякую за увагу!